A TRIDENT SCHOLAR PROJECT REPORT

NO. 247

CONTINUOUS LOCALIZATION AND NAVIGATION OF MOBILE ROBOTS

UNITED STATES NAVAL ACADEMY
ANNAPOLIS, MARYLAND

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATE COVERED
	7 May 1997	

4. TITLE AND SUBTITLE
Continuous localization and navigation of mobile robots

5. FUNDING NUMBERS

6. AUTHOR(S)
Kevin P. Graves

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Naval Academy
Annapolis, MD

8. PERFORMING ORGANIZATION REPORT NUMBER
USNA Trident Scholar project report no. 247 (1997)

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Accepted by the U.S. Trident Scholar Committee

12a. DISTRIBUTION/AVAILABILITY STATEMENT
This document has been approved for public release; its distribution is UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT: A large mobile robot was used as a platform for research in continuous localization and path planning. Continuous localization is a technique that allows a robot to maintain an accurate estimete of its location by performing regular, small corrections to its odometry. Continuous localization utilizes an evidence grid representation, a common representation scheme that is used by other map-dependent processes, such as path planning. Although techniques exist for building evidence grid maps, most are not adaptive to changes in the environment. In this research, the continuous localization technique is extended by adding a learning component. This allows continuous localization to update the long-term map (evidence grid) with current sensor readings. Results show that the addition of the learning behavior to continuous localization allows the system to adapt to changes in its environment without a loss in its ability to remain localized. Continuous localization with the learning behavior was combined with a wavefront propagation path planner to produce a robust navigation system. This system was tested on a Nomad 200 mobile robot.

14. SUBJECT TERMS
mobile robots, continuous localization, evidence grids, wavefront propagation

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

U.S.N.A. --- Trident Scholar project report; no. 247 (1997)

CONTINUOUS LOCALIZATION AND NAVIGATION OF MOBILE ROBOTS

by

Midshipman Kevin P. Graves, Class of 1997
United States Naval Academy
Annapolis, Maryland

Kevin P. Graves

Certification of Adviser Approval

Professor Patrick R. Harrison
Department of Computer Science

Patrick R. Harrison
8 May 1997

Acceptance for the Trident Scholar Committee

Professor Joyce E. Shade
Chair, Trident Scholar Committee

J E Shade
8 May 97

USNA-1531-2

Abstract

A large mobile robot was used as a platform for research in continuous localization and path planning. Continuous localization is a technique that allows a robot to maintain an accurate estimate of its location by performing regular, small corrections to its odometry. Continuous localization utilizes an evidence grid representation, a common representation scheme that is used by other map-dependent processes, such as path planning. Although techniques exist for building evidence grid maps, most are not adaptive to changes in the environment. In this research, the continuous localization technique is extended by adding a learning component. This allows continuous localization to update the long-term map (evidence grid) with current sensor readings. Results show that the addition of the learning behavior to continuous localization allows the system to adapt to changes in its environment without a loss in its ability to remain localized. Continuous localization with the learning behavior was combined with a wavefront propagation path planner to produce a robust navigation system. This system was tested on a Nomad 200 mobile robot.

Keywords: mobile robots, continuous localization, evidence grids, wavefront propagation

Table of Contents

1.0 Introduction

1.1 Motivation

In countless novels and movies, robots imitate intelligent human behaviors such as bipedal locomotion, speech, and insight. These truly autonomous mobile robots, however, currently exist only in science fiction. Much improvement is needed before robots will be able to walk, interactively communicate, and think completely on their own.

The potential uses for such robots are unlimited. Modern applications, however, can be satisfied with robots of a more limited nature. The Navy needs autonomous robotic systems that can be deployed in remote and hazardous environments. Potential Navy applications of autonomous systems include shipboard fire fighting, hazardous material handling, surveillance, salvage, and undersea equipment maintenance.

A truly autonomous robot currently does not exist. Before high-level tasks can be carried out by mobile robots, robot systems must incorporate four basic abilities. First, a truly autonomous robot would create a map of its environment without *a priori* information and subsequently updates the map when changes in the environment are identified. Second, the robot always would know its location and orientation relative to the map. Third, the robot system would devise efficient, robust paths to goal locations. Fourth, while traveling to the goal locations, the robot would avoid any unexpected obstacles while proceeding along the most efficient path to the goal. These four abilities comprise the "robot problem," for which a complete solution has not been devised.

This study focused on improving two abilities of autonomous mobile robots. The robot was provided an *a priori* map of its environment. Once an optimal path to the next goal location was devised, the robot employed a simple, non-optimal collision avoidance strategy. This project contributes to the design of future autonomous systems by extending recent results in continuous localization and mobile robot navigation.

1.2 Overview

An important feature of mobile robot systems is the ability to operate in dynamic environments without degradation in performance. In other words, the robot systems must adapt the robots' behaviors to account for any changes which occur in the environment. Examples include a person walking near a robot or, for a fire-fighting robot onboard ship, a missile striking the ship and creating a hole in the bulkhead of a passageway through which the robot normally travels.

For mobile robots to perform autonomously in dynamic environments, they first need the ability to determine their location in the world. This includes the ability to learn maps of their environment and to use these maps to localize themselves over an extended period of time. Localization is the process of determining the robot's pose (position and

orientation) relative to its surroundings.

A map is a robot's model of its world. Maps are required for the robot to perform tasks, such as moving two feet forward or planning the most efficient path to a goal, and to allow the robot to reason about the spatial characteristics of its environment. Most mobile robot systems provide the robot with an *a priori* map so that the robot will have an estimate -- sometimes an exact representation -- of its environment. When an *a priori* map is not available, the robot must learn a map by recording the locations of objects the robot senses.

While many researchers have addressed map learning, most have not addressed methods to update maps to reflect changes in a dynamic environment. The researchers assume the room will remain static. In the real world, however, environments are dynamic and do change over time. Since most robot processes use a map, it is imperative to maintain a map that accurately models the environment.

Simple localization techniques, such as dead reckoning, can place the robot within its map, but do not provide accurate and reliable localization estimates. Dead reckoning records the movement of the robot's wheels in order to determine the current pose relative to a known starting point. This process is similar to the inertial navigation used by airplanes and missiles. Relying solely on dead reckoning causes a robot to veer considerably off course over time because odometric errors accumulate quickly. Odometric error is the difference between the robot's odometry (the pose it thinks it holds) and its actual pose. Odometric error is caused primarily by the intermittent slippage of the wheels in the course of normal operation. Other factors include slippery spots, dirt on the floor, and uneven surfaces, all of which can cause one wheel to spin faster than the others.

More advanced localization techniques typically depend on recognizing, and in some cases learning, individual landmarks in the environment (Thrun, 1996). Landmark recognition, which often is used in conjunction with dead reckoning, is more accurate than dead reckoning alone. Natural landmarks (i.e. walls or surfaces that make up the perimeter of the permanent region) or artificial landmarks (such as beacons or other objects placed in a region only to be used as navigation aids) are observed with cameras, infrared imagers, ultrasound, or other sensors. But landmark navigation also relies on the navigation aids remaining static. If the supposedly fixed landmarks move, then the robot will not know its true location, nor have a means of obtaining it.

Researchers have looked at using the same map for their localization technique and navigation scheme (Yamauchi et al., 1997a). Most localization schemes are fairly robust, even when changes in the environment are not recorded in the world map. However, the ability to localize or navigate will degrade when the world map does not model substantial changes to the environment.

Continuous localization (Section 4.2) uses a robot's local environment, the area within the range of its sensors, to periodically correct its odometry error on the fly. Continuous localization has been demonstrated to eliminate the accumulation of odometry error while maintaining a constant translational error on the order of five inches (Schultz, et al., 1996).

The continuous localization method uses an evidence grid representation to model its environment (Elfes, 1992; Moravec, et al., 1985; Hughes et al., 1992).

Evidence grid representations have also been used for navigation, path planning, and other map-based tasks. Using the same representation in both the localization and other processes allows for a more uniform representation and a simpler system.

In previous work, continuous localization utilized *a priori* evidence grid maps of the environment. Other recent work has looked at learning the initial evidence grid maps through an exploration strategy (Yamauchi et al., 1997b). These learned maps can then be used as initial maps by continuous localization.

In this research, the continuous localization technique is extended so that it allows the long-term map of the environment to adapt to changes in the environment by incorporating incoming sensor data. The results indicate that the addition of the ability to learn dynamic changes in the environment does not degrade the performance of localization, and at the same time yields a more accurate long-term map for other processes that use the map.

Once a robot knows its pose precisely, it can perform higher-level functions such as path planning and navigation. Most mobile robot systems utilize a deliberative-reactive architecture. An explicit path planner (the deliberative portion) computes a route to the goal location specified by the human user. Ideally, the path planner will produce an optimal (most efficient) route. The navigation scheme uses this route as a guide; the robot will follow the path to the goal until it encounters an unmodeled obstacle, in which case it will reflexively avoid a collision.

The potential field method (Section 4.3.1) is an example of a non-optimal and error-prone path planner (Arkin, 1989). The robot models its world much like a gravitation field. The goal location exerts an imaginary attractive force on the robot while obstacles repel the robot. A rasterized grid stores the sum of all imaginary forces acting on every point in space. The robot moves to the adjacent cell with the greatest net attractive force. Hence, the robot follows the least-cost path to the goal, much as water flows down hill or a positive charge is drawn toward a negatively charged magnet. However, potential fields do not provide robust solutions. They can direct the robot into a modeled obstacle. Or, if the sum of forces at a position other than the goal equals zero, the robot might navigate to this position and stop.

Wavefront propagation (Section 4.3.2) is an improved form of potential field representation. Trulla is a wavefront propagation path planner that provides robust routes to the goal (Hughes et al., 1992). Trulla's solution is a vector field that tells the robot, from every point in modeled space, the optimal direction to travel. By planning every optimal path for every position beforehand, when the robot reactively avoids an unmodeled obstacle, it will find a new optimal path to the goal. In some situations, unmodeled obstacles will completely block the path planner's optimal routes. After using the robot's sensors to learn these obstacles, the path planner can be re-invoked. The resulting optimal paths will direct the robot around the now-modeled obstacles.

Current implementations of Trulla possess problems. First, at approximately two inches per second, the speed at which the robots travel is extremely slow (Murphy et al., 1997). Secondly, the robot system lacks an accurate localization technique. By relying on dead reckoning, the robot's odometric error accumulates over time.

This study solved these problems. By incorporating continuous localization (with learning behavior) and a wavefront propagation path planner, unmodeled obstacles

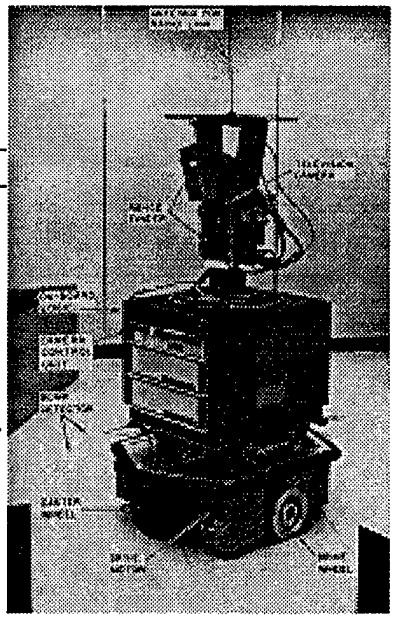
were assimilated into the map and a new optimal path was computed on the fly. The author's system maintains an accurate update of the robot's position while moving at five times the speed of Murphy's robot.

2.0 History of Mobile Robots

Primitive robots were controlled remotely by human operators and did not possess the ability to maneuver themselves. The addition of a television camera on a robot allowed humans to control the robot beyond visual range. The inherent disadvantage of these remotely operated vehicles (ROVs) is evident in interplanetary explorers. Even at the speed of light, a signal sent from Earth takes minutes to travel to even the closest planet. Improvements in sensors, the efficiency of algorithms, and sheer processing power have made mobile robots more independent of human control. However, even today's robots are not truly autonomous. Although the robot's computer performs all of the calculations and controlling schemes, humans still manipulate the robot by loading *a priori* maps into the robot's memory or making artificial adjustments to the environment (such as radio beacons or objects from which a visual reference can be obtained). In the artificial intelligence community, "autonomous" refers to a robot whose programming allows it to navigate from a starting point to a goal without remote control while recognizing and overcoming unpredictable situations. This project takes one more step toward the creation of a truly autonomous robot.

2.1 Shakey

Shakey was the first attempt at an autonomous mobile robot. Completed at Stanford Research Institute in 1969 and used until 1972, Shakey employed the premiere technology of the time: a television camera, a range finder, and a radio link to a powerful offboard computer which processed the vision and planning elements. The onboard computer only had the capacity to control the motors.



Shakey (Moravec, 1981)

Dead reckoning, enhanced with a uni-camera vision system and a range finder, provided for Shakey's localization. Shakey determined its position by first locating the boundary between floor and wall (in the image provided by the camera). Shakey then compared the length and angle of that line segment with a predicted length and angle. The predicted values were determined from the world model Shakey kept in memory (i.e. what Shakey thought it should see based on the stored map and pose estimate). The angular discrepancy between actual and predicted values was used to correct Shakey's pose estimate. Since the entire process took over 10 seconds, Shakey could not be moving when it localized or else it would have updated its pose based on an outdated position.

The uni-camera system's range error was 5-10%, and the angular error was 5% (Nilsson, 1984). This error was due to mechanical constraints of the camera as well as shadows and reflections.

Figure 1 shows a representation of Shakey's path analysis approach. Shakey's path planner analyzed the world map created from the images provided by the camera. Objects were modeled as circles. All of the possible routes from robot to goal that traversed the tangents of the obstacles were recorded onto a tree data structure. To minimize computing time, the chosen path was determined by a breadth-first search of this tree. Thus, Shakey travelled the path with the fewest legs (line segments) rather than the shortest distance (Nilsson, 1984).

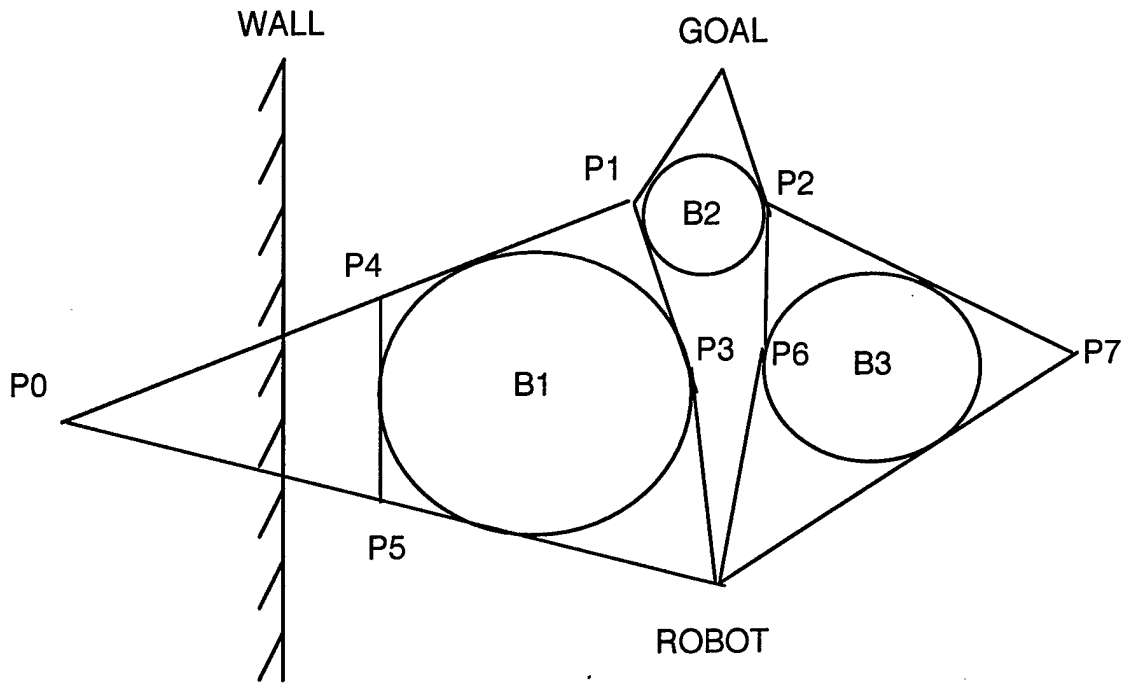
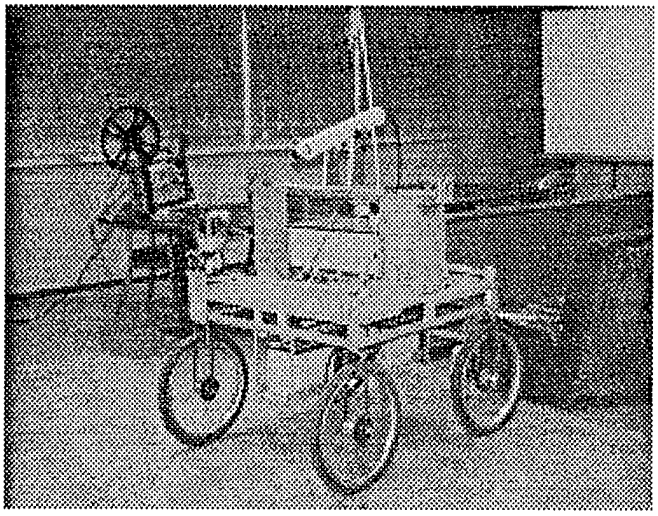


Figure 1: Shakey's Path Planner (Nilsson, 1984)

Although Shakey was a tremendous advance in robotics, the inaccuracies of both its dead reckoning and vision system hindered performance. Shakey's odometry error was so large that in complex environments the distances and angles of the line segments from its images could not be matched with the predicted values from the world map (Meystel, 1991). In these situations, the robot did not find the goal.

2.2 The Cart

Hans Moravec, one of the pioneers in the field of autonomous mobile robots, worked on the Cart at the Stanford University Artificial Intelligence (AI) Lab from 1973-1981. This research vastly improved the use of vision as a navigation tool. The television camera mounted onboard the Cart allowed remote control. It also gave the robot the ability to make maps and navigate without direct human guidance. Though slow (in both physical speed and processor speed), the Cart was able to make an accurate internal model of its environment and successfully navigate around and in-between objects. Depth perception was achieved by imitating the way in which a lizard bobs its head back and forth while assessing the fly it is about to catch with a tongue flick. Every time the Cart stopped to localize, the camera took a picture from each of nine angles by sliding the camera along a track (Moravec, 1981). The work was continued at Carnegie-Mellon University (1981-1984).



The Cart (Moravec, 1981)

2.2.1 Drawbacks of Cameras and other Sensors

Uni-camera systems such as Shakey and the Cart had numerous limitations. After taking a picture, the frame was represented as a rasterized grid (Figure 2). Each cell was assigned a number from zero to sixty-three to represent the shade of the object, with lower numbers corresponding to lighter shades. By grouping cells of similar shade the robot identified objects and consequently determined distances and angles. The two primary drawbacks of this method of navigation are obvious: shadows and reflections confused the camera's limited depth perception, and objects of the same color were difficult to differentiate.

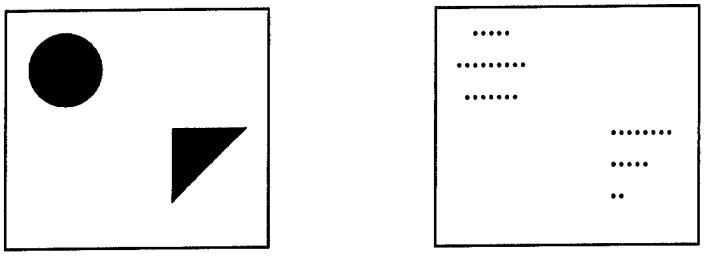


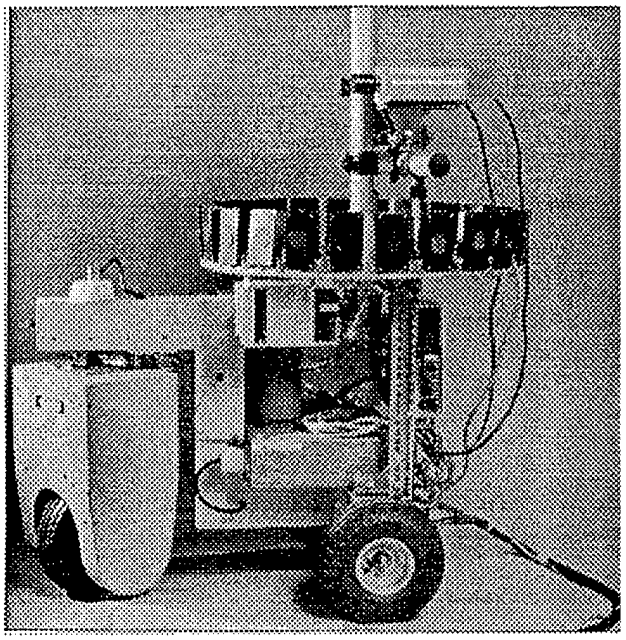
Figure 2: Example of a region and its corresponding rasterized map

Additional sensory inputs were added to battle the limitations of single cameras, though robust solutions still were not achieved. Laser range finders were mounted alongside cameras in order to improve distance measurements. Although they are accurate to a distance of 0.5 inches, each reading only provided the distance to a particular point in space. In order to scan an area 5 feet by 5 feet, it would have taken a laser range finder 7 seconds. Although this sounds like a short time period, if the robot or the object is moving, the delay would have caused the robot to think the object was many feet from its actual location.

Stereo vision -- two cameras with overlapping fields of view -- yielded improved depth perception. However, it required lengthy computation time and thus reduced the number of points that could be tracked. This in turn degraded the detail of the produced map (Elfes, 1987). Moravec's fastest computers took 30-60 seconds to process a low-resolution image, during which the Cart traversed one meter (Moravec, 1984). A system of localization was needed that provided accurate distances to objects and required low-cost hardware and little computation.

2.3 Neptune

Moravec's next robot, Neptune, relied on sonar to overcome the inherent errors and lengthy computation time of vision-based systems. Twenty-four wide-angle sonars provide 360 degree coverage (Moravec, 1984). As the beam width of each sonar was 30 degrees, the overlapping coverage guaranteed that almost the entire area in the path of the sonar wave would be seen by at least two sonars. The transducers that heard the ultrasonic echo could identify objects out to thirty-five feet. (See Section 4.1.1 for a thorough description of the operation of sonars.)



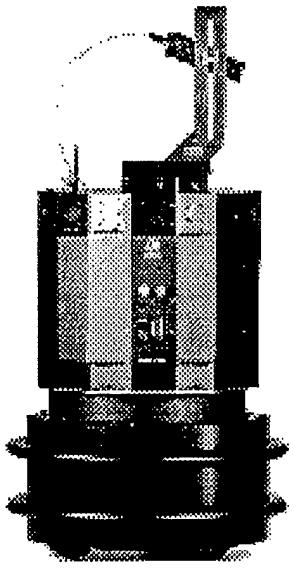
The Neptune (Elfes, 1987)

Neptune was the first robot to incorporate the evidence grid representation (Section 4.1) to model its world.

2.4 Coyote

The Navy Center for Applied Research in Artificial Intelligence (NCARAI) maintains two Nomad 200 robots, Coyote and Roadrunner. All experiments described in this paper were performed using Coyote.

Coyote is equipped with four groups of sensors. In the picture at right, the two bumpers can be seen encircling the robot's cylindrical base. When Coyote runs into an object with sufficient force to trigger a bumper, all movement ceases. Ideally, the navigation scheme will avoid all objects.



Coyote

The sixteen-sided turret, located above the base, holds two groups of sensors. Sixteen infrared sensors (which are too small to be visible in the picture at right) circle the bottom of the turret and sixteen sonars circle the top. Infrared sensors emit light in the frequency range 10^{12} to 10^{14} Hz. The sensors detect the intensity of the light reflected off of an object and have a maximum range of approximately sixteen inches. The reading, a number between 1 and 16,

approximates the distance to the object in inches. Infrared readings are not reliable estimates of distance because the color, texture, and material of the object affect the intensity of the reflected light. The infrared sensors' angular range is 20 degrees. Coyote's sonars sense objects within a 25 degree cone out to a range of approximately thirty-five feet.

Coyote's fourth sensor, a structured light range finder, resides above the turret. The box mounted directly to the turret works in conjunction with the camera (the cylindrical object with the wire connecting it to the turret). The laser and camera remain fixed to the turret. The turret turns so that the sonars and laser can point directly at any object. Except when backing up to avoid an obstacle, Coyote typically moves only in the direction the camera faces. A laser travels through a cylinder within the box and is emitted as a horizontal plane at an angle of 30 degrees. The human eye (and the camera) can see a red line where the laser plane strikes an object. The camera records the image it sees at a rate of 30 Hz. The laser line is modeled as 240 individual points (the width, in pixels, of the image). Since the camera and laser are in fixed positions relative to each other, each pixel corresponds to a constant distance. A point higher in the image equates to a greater distance. Thus the robot determines two-dimensional range data simply by determining the laser point's pixel coordinates and using geometric equations to convert that value to a distance. The structured light range finder is accurate to within one millimeter at a distance of twelve feet.

An Intel 120MHz Pentium processor comprises the onboard computer. It operates using LINUX, a UNIX operating system. The onboard computer continually runs a robot daemon, which is an interface between the program and the actual signals to the robot's motors and from the robot's sensors. The daemon constantly checks for new motor commands and immediately executes them. However, the programs written for the robot do not normally reside on this computer. Instead, they are stored and executed on a Sun workstation. The commands are delivered to Coyote via a wireless ethernet connection with the robot. Occasionally, a program will reside on the robot's computer. Since no connection with a network exists, these programs execute faster than if they were to be run from a Sun workstation. Examples of such programs include video processing (because of the intensive amount of data that otherwise would be transmitted across the network) and collision avoidance (which requires immediate response to avoid obstacles in the robot's path). Most programs do not require the improved performance from being run on Coyote.

Coyote employs five lead-acid batteries, two to power the motors and three for the computer. These batteries need to be recharged approximately every 100 minutes.

3.0 Modern Robotic Systems

3.1 Evidence Grid Representation

3.1.1 The Operation of Sonars

The evidence grid representation originally was created to take advantage of sonar sensors. As of 1984, sonar provided an inexpensive means of determining ranges to nearby objects, though few people had attempted to build maps from sonar readings (Moravec, 1984). Sonar waves are emitted as a spherical wavefront. However, since most implementations use only two-dimensional maps, the wave is commonly represented as a two-dimensional cone (Figure 3). The cone's angle of coverage (angular range) depends on the receiver's sensitivity. Range readings are determined when an echo (a reflection off of an object) is detected by the emitting sonar's transducer. It is imperative to understand that a transducer records only the first echo it receives from each wave emitted; all subsequent data are ignored. Half of the time of travel multiplied by the speed of the wave yields the distance to the object.

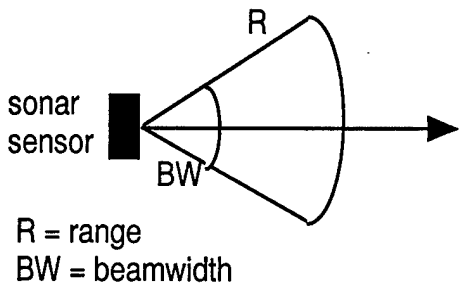


Figure 3: Volumetric Cone of a Sonar Beam

Notice that the exact direction to the object that caused the echo is unknown. Each reading yields a range to the closest object within the sonar cone. The position of the sonar on the robot and the sonar's angular range provide merely an estimate of the object's location.

Only a fuzzy image of the perimeter of objects can be constructed from sonar readings because of specular reflection and the attenuation of ultrasonic waves. Specular

reflections occur when a sonar beam does not reflect directly back to the sensor; the range reading is greater than the actual distance to the closest object (Figure 5). Thus specular reflections cause the robot to record on its map that it detected an object at a position much further away than the object's true location. Natural attenuation of an ultrasonic wave results from a weak signal, usually when an object lies near the edge of the volumetric cone (the least sensitive portion of the receiver). For narrow-beam (1-3 degrees) sonar, the inherent error is minimal. However, to implement narrow-beam sonar, 120-360 such sonars are necessary. This is impractical because of the inordinate computation time needed to scan all 360 degrees around the robot. Wide-angle sonars are preferred, even though their waves attenuate more than narrow-beam ultrasonic waves. Overlapping the volumetric cones of wide-angle sonars reduces the number of sonars and thus the computation time (Figure 4).

Several characteristics of sonar cones are illustrated in Figure 6. The values represent the evidence of occupancy for that cell. A greater value corresponds to a greater probability of occupancy. The first observation from Figure 6 is that a single wide-angle sonar reading possesses inherent error. Occupancy values are assigned to cells that obviously are not occupied. Secondly, sonars are reliable at identifying empty space. In this example, every space that holds a value of 0 actually is empty. Due to the equations governing sonar readings, evidence that indicates emptiness is more dependable than values that indicate occupancy. The fringes of occupied space provide the most unreliable readings. Third, overlapping empty and occupied volumes reinforce the evidence provided by each other. Also, the evidence of a single specular reflection is reduced when averaged with multiple correct readings. In Figure 6, the overlapping sonars in Part B provide more accurate readings than the single sonar in Part A.

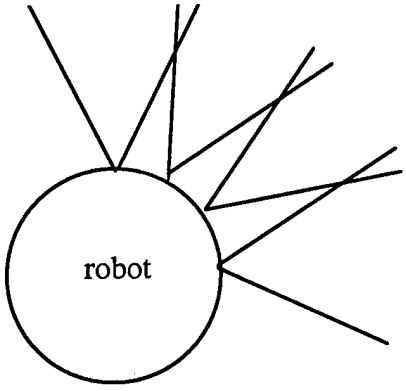
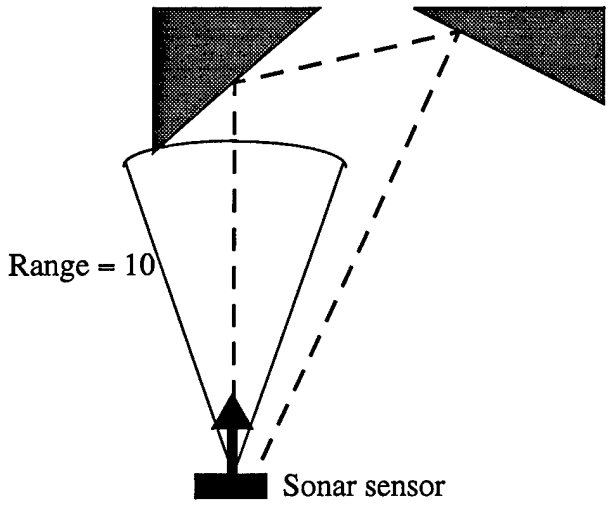


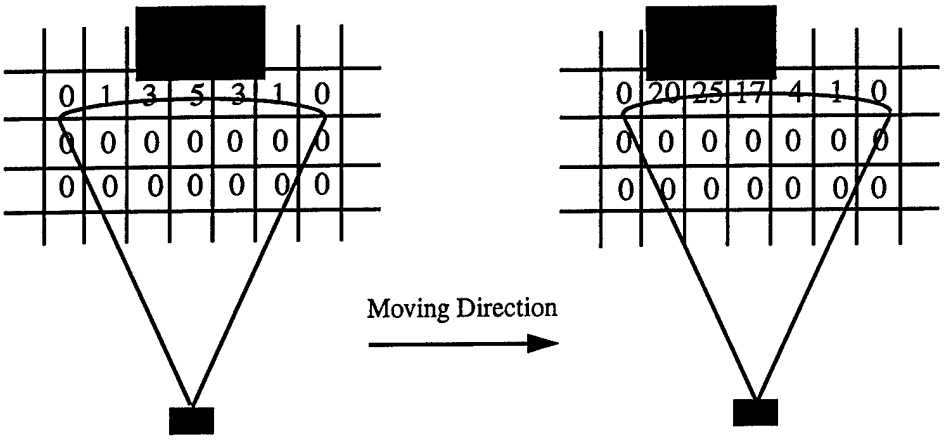
Figure 4: Overhead view of overlapping sonars



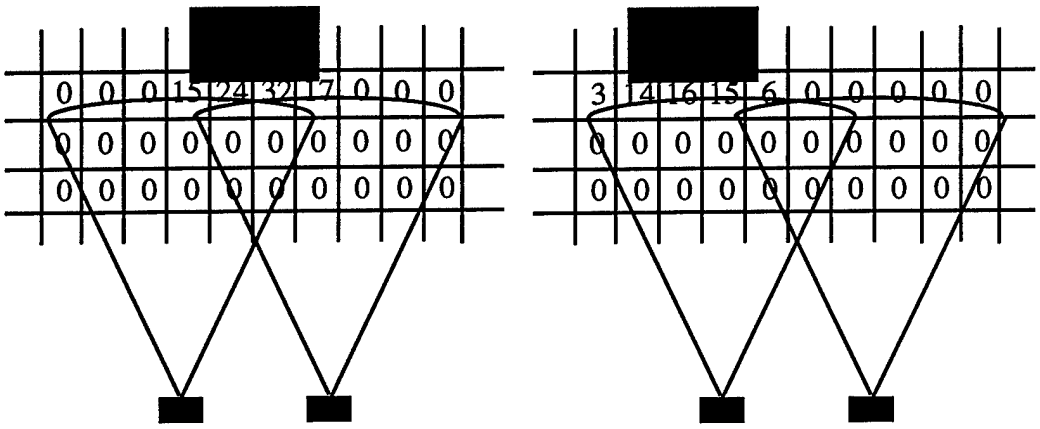
—— actual range to closest object = 10

- - range reading (i.e. the distance the wave actually traveled) = 26

Figure 5: Specular Reflection



Part A: Sonar readings in motion reinforce certainty values.



Part B: Overlapping volumetric cones reinforce occupied and empty values.

Figure 6: Overlapping Sonar Cones (Chung et al., 1992)

3.1.2 Evidence Grids

Moravec and Alberto Elfes (1984) made a breakthrough in localization theory by inventing a method of representing ultrasonic data that overcame sonar's inherent error. Since each individual sonar reading provided a range and a rough estimate of direction, Moravec mapped the sensory data statistically. Probability equations modeled the reliability of each reading. Although a single sonar reading provides inconclusive data, combining hundreds of sonar readings accumulated during one second of time yielded a fairly accurate and reliable map (Moravec, 1984).

Moravec's new method modeled Neptune's room with a rasterized grid. One cell on this map corresponded to a 30cm by 30cm square in the room. Each cell was assigned two numbers -- the degree of certainty that the cell was empty (Emp(X,Y)) and the probability that the cell was occupied (Occ(X,Y)) -- that were updated with each sonar reading. Both numbers ranged the interval [0, 1], and were initially set at 0 (unknown, or no evidence to suggest one way or the other). Due to the nature of Moravec's methods for combining evidence, the probability that the cell was occupied and the probability that the cell was empty do not necessarily sum to a value of one. For example, if cell (2,3) contained the values Emp(2,3) := 0.0335 and Occ(2,3) := 0.8842, you would be quite confident that the cell is occupied.

Moravec processed the raw data to correct for erroneous readings. He discarded any readings beyond an experimentally determined threshold (which was slightly less than the maximum range of the sonar) to reduce specular reflection. Fluctuations in the transducer's sensitivity were minimized by disregarding readings below the minimum range of the sensor. Also, since natural errors caused a dispersion of readings taken from a single sensor at the same position, Moravec averaged all data received while the robot remained stationary.

To account for the inaccuracy of the readings, a stochastic sensor model determined the readings' influence on each cell. The sonar cone was modeled by probability distribution functions (Appendix A). The equations for combining multiple sensor readings are found in Appendix B. These statistical functions, also referred to as probability density functions, represented the degree of certainty that the points inside the beam's cone were empty (since no echo was returned from them) as well as the uncertainty of the exact location of the point which reflected the beam.

The values of Emp(X,Y) and Occ(X,Y) for each grid cell within the volumetric cone were entered into these equations. The solution was a single value (for each grid cell) in the range (-1, 1) that represented the overall probability of that cell's occupancy. This result was added to the value already stored in that cell.

One of Moravec's final evidence grids is given in Figure 7. A negative number denoted a probably empty point and was represented on the final map with white space. An increasingly negative value correlated to more evidence that the point in space was empty. Likewise, a positive number signified a probably occupied point and was charac-

terized by a '+'. A larger '+' corresponded to a greater probability of occupancy. A value near 0 indicated that the occupancy of the cell was unknown. Either no evidence at all was obtained about that point (e.g. it was behind a wall and thus beyond reach of the sonar beams) or the cumulative sum of all evidence totalled zero. An unknown region was denoted by a '.'.

The most significant aspect of Moravec's representation is that the use of Bayesian statistics bestows evidence grids with the ability to fuse readings from any sensor. Robots equipped with a variety of sensors could utilize Moravec's evidence grid representation to combine sensor data on one map used by all robot processes.

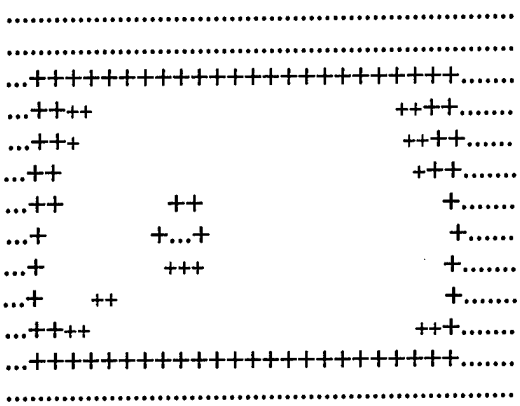


Figure 7: Example Evidence Grid

3.2 Continuous Localization

The evidence grid representation provides a robust means of mapping a mobile robot's environment. However, in order to use this representation, the robot must be localized. Moravec intended Neptune to be placed in an unknown environment without *a priori* information on the region where it would create an evidence grid with which to navigate accurately. But the odometry error introduced into the maps degraded pose estimates over time.

Horn and Schmidt (1995) innovated a method of correcting odometry error by integrating single-image localization and dead-reckoning data approximately every 2.4

seconds. They named this technique continuous localization. Previous localization methods localized less often, usually only after the robot's uncertainty of odometry error passed a threshold. Single-image localization was achieved by matching "vertical planar surfaces extracted from a 3D-laser-range-image with corresponding surfaces predicted from a 3D-environmental model" (Horn et al., 1995). However, this approach required an *a priori* map and did not model uncertainties in the map.

The Navy Center for Applied Research in Artificial Intelligence (NCARAI) simultaneously developed a different method of continuous localization which modeled the map's uncertainties. Although their method also needed an *a priori* map, the use of evidence grids allows the robot to fuse data from all types of sensors incrementally as well as model slow changes in the environment.

Any continuous localization technique requires the periodic registration of local perceptions with some description of the environment (a map). The first important issue to be considered is, therefore, the representation used to store the map. The second issue is how often to perform the registration.

Continuous localization uses evidence grids for its representation because they provide a uniform framework for fusing temporally and spatially distinct sensor readings. All robot sensors contribute to the task of localization, and the system is robust in the face of sensor failures and noise in individual sensor readings.

The evidence grid representation divides a volume into equally sized cells. Each cell contains a single real number between -1 and +1 that describes the evidence of that cell being occupied. A value near +1 suggests reasonable assurance that the cell is inhabited, while -1 signifies substantial evidence that the cell represents empty space. Cells are initialized to 0 (unknown).

Cells are updated from sensor readings that are filtered through Moravec's probability distribution functions. After each sensor reading, all relevant cells are updated using the new evidence from the sensor. Several methods have been used to update the evidence in the cells including Bayesian statistics (Elfes, 1992; Moravec, 1988) and Dempster-Schafer theory (Hughes et al., 1992). In the work presented here, the Bayesian update method was used.

The second issue in localization, how often to relocalize the robot within its environment, is addressed in many techniques by only relocalizing when either an error in position is detected or after an unacceptable level of odometry error has accumulated. Continuous localization, however, makes regular, incremental corrections to the odometry rather than intermittent corrections requiring expensive computation. Since the pose error accumulated during the time interval between corrections is small, the registration search can be restricted to a small area around the current pose, reducing the computational effort required and sustaining the higher relocalization frequency.

Continuous localization utilizes an *a priori* long-term map which is assumed to be an evidence grid representation of the region (room) in which the robot is currently operating. The robot builds a continual series of short-term perception maps of its immediate

environment, each of which is of brief duration and contains a small amount of pose error. After several time intervals, the oldest (most "mature") short-term map is used to position the robot within the long-term map via a registration process.

The registration process consists of sampling the possible poses within a small area around the robot's current pose. Small adjustments in position and orientation are made to the values of the current odometry, but the robot does not actually move to the new pose. For each tested pose, the mature short-term map is adjusted by the offset between the hypothetical pose and the robot's actual pose. A match score for each pose is calculated based on the number of cell values of the short-term map which are equivalent to their corresponding cell values on the long-term map. The match scores for all tested poses are then treated as masses and the offsets as distances, and a center of mass calculation is performed to determine the offset that corresponds to the highest match score. This offset is applied to the robot's odometry, placing it at the pose which causes its local perceptions to best match the long-term map. All robot processes subsequently use this new odometry. After the registration takes place, the most mature map is discarded.

In this research, the continuous localization algorithm has been extended to allow the long-term map to be updated with current sensor data from the short-term perception maps, thereby making the long-term map adaptive to its environment.

Rather than immediately discarding the most mature short-term map after registration, the short-term map is corrected in pose, weighted, and its cells combined with the spatially corresponding cells of the long-term map. The weight applied to the short-term map is a learning rate that controls how much effect the short-term map has on the long-term map. The learning behavior can be triggered on or off at runtime.

3.3 Path Planners

Continuous localization provides a robot with the ability to correct its odometry. However, in order to travel intelligently, a robot needs a separate process to plan a path from its current location to a specified goal. Continuous localization merely tells the robot its pose as it travels along a path.

3.3.1 Potential Fields

Ronald Arkin and Robin Murphy (1989) designed potential fields to serve as an optimal path planner. Although potential fields can be used by any sensor type to construct a world map, its method of modeling the world is too simplistic. Many situations exist where potential fields will not plan an optimal path to the goal.

The potential field representation models the world much like a gravity field. Obstacles exert imaginary repelling forces while the goal location attracts the robot. A

rasterized grid stores the sum of all imaginary forces acting on every cell. In mathematical terms, the repulsive force (F_{xy}) for each cell (x, y) is proportional to the probability of occupancy of the cell and inversely proportional to the distance of the cell from the center of the robot. The attractive force (F_g), though related to the distance from the goal to the center of the robot, largely depends on an experimentally determined constant. The attractive force added to the sum of (F_{xy}) yields the resultant force (F_R) (Koren et al., 1991). The robot moves to the adjacent cell with the greatest net attractive force. Hence, the robot follows the least-cost path to the goal, much as water flows downhill.

Three fundamental problems of potential fields have been identified. First, a robot may become trapped in a local minimum. A local minimum exists where the resultant force at every adjacent cell directs the robot toward the cell the robot currently occupies. Therefore, the path planner cannot compute a path away from the cell. Thus the robot stops moving because it thinks it has reached the goal (Figure 8). Second, two obstacles can be spaced such that a navigable passage between them exists, but their repulsive forces overlap and do not permit passage. Third, a robot travelling through a narrow corridor will receive repulsive forces from both walls simultaneously. This will result in an oscillating motion; although the robot will advance through the corridor, much time will be wasted following a winding path. (Koren et al., 1991).

The most commonly cited example that demonstrates potential fields' shortcomings is the box canyon (Figure 8). Although the obstacle exerts a repelling force on the robot, the greater force from the goal attracts the robot into the canyon. Once in the canyon, potential fields offer no hope of escape because the attractive force does not let the robot move far enough away from the goal to exit the canyon. A separate scheme (usually a random motion generator) is needed to override the potential fields.

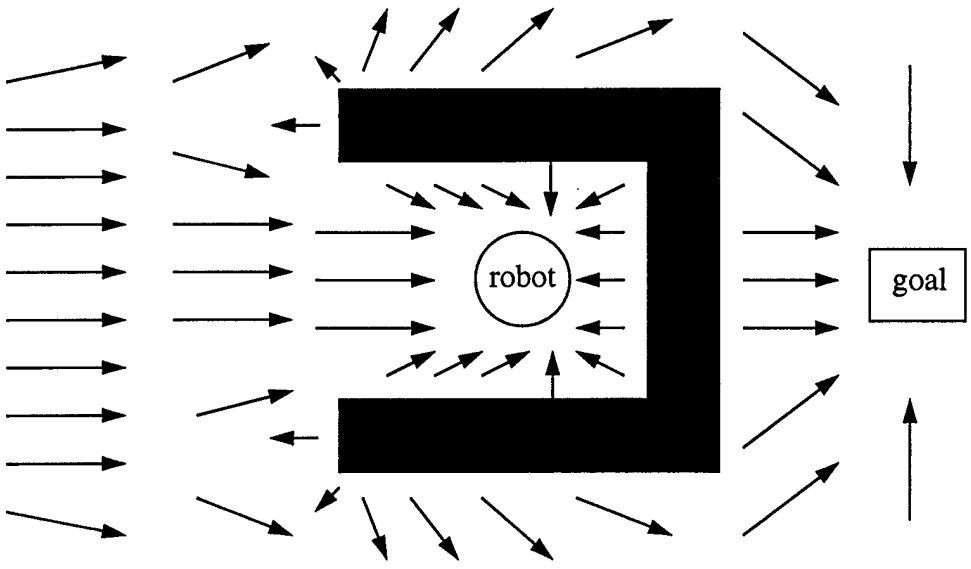


Figure 8: Example of Local Minimum Caused by a Box Canyon

3.3.2 Trulla

Path planning is a computationally expensive task. Ken Hughes (1992b) devised the Trulla algorithm to reduce the cost of a path planner by taking advantage of parallel architecture.

Wavefront propagation techniques are characterized by their ability to compute a near-optimal path to the goal by allowing neighboring cells on a rasterized map to share with each other information regarding the best path to the goal.

The Trulla algorithm possesses several improvements over previous wavefront propagation methods. First, it performs simple computations on local data. Second, Trulla allows repeated propagations which converge to a solution. Third, the algorithm computes near-optimal paths from all locations to the goal instead of one optimal path from the robot's current position to the goal. This last factor makes Trulla ideal for a deliberative/reactive architecture. By computing the set of all near-optimal paths beforehand, reaction to an unmodeled obstacle does not require recomputation since the robot is merely displaced to another near-optimal path (Murphy et al., 1997b).

Hughes's Trulla implementation possesses a few drawbacks as well. It assumes that the robot operates in a static environment and depends on an *a priori* map. The map contains a weight for each cell that indicates traversibility, or the effort the robot must expend to navigate through that cell. A cell with a cable across the floor has a higher weight than a cell which is completely empty. Moreover, Trulla only allows eight-neighbor connectivity. In other words, only eight possible angles (corresponding to the eight cardinal directions) can be given to the robot.

For every cell in which a direct path cannot be executed to the goal, a subgoal is established. The heart of the Trulla algorithm is a set of rules that determines when a path can be propagated to its neighbors unaltered and when a new subgoal must be established. Trulla also must determine, at each cell, whether the current best path should be replaced by a best path from a neighboring cell.

Robin Murphy (1997b) improved upon the Trulla algorithm. Her work assumes operation within a dynamic environment and has the ability to update its map when the robot's sensors detect unmodeled obstacles. Also, the resultant vector (direction and velocity) assigned to any grid cell can be at any angle (as opposed to eight-neighbor connectivity) (Figure 9). Darker shades correspond to a higher weight of traversibility.

Murphy's implementation also loads the weights *a priori*. Each grid cell on the robot's map has dimensions equal to the diameter of the robot. An entire cell is considered to be occupied if an object inhabits any part of the cell. This excludes a path from being propagated through a corridor narrower than the robot (except when unmodeled obstacles are present).

Unmodeled obstacles may trap a robot by creating a situation similar to a box canyon. In such a configuration, the robot system must learn the location of the obstacles and replan a path around them. Much research has been devoted to how often the robot system should replan. Continuous replanning is best, if computational expense is not a factor (Murphy et al., 1997b).

Murphy tested her system against four scenarios: (1) an unmodeled box canyon, (2) an unmodeled wall (where the robot can travel on either side to get to goal), (3) part of a wall modeled *a priori*, with a portion of the wall (in the direction Trulla would send the robot) unmodeled, and (4) an unmodeled wall similar to scenario 2, except that the wall intersects another wall perpendicularly and hence is untraversible on that side.

The algorithm boasts a replanning time that never exceeds .94 seconds. This is due to the parallel architecture of the robot system and the large grid size. However, during the experiments, the robot moved at approximately two inches per second! Murphy's results may not be applicable to a robot moving at a reasonable speed (Koren et al., 1991).

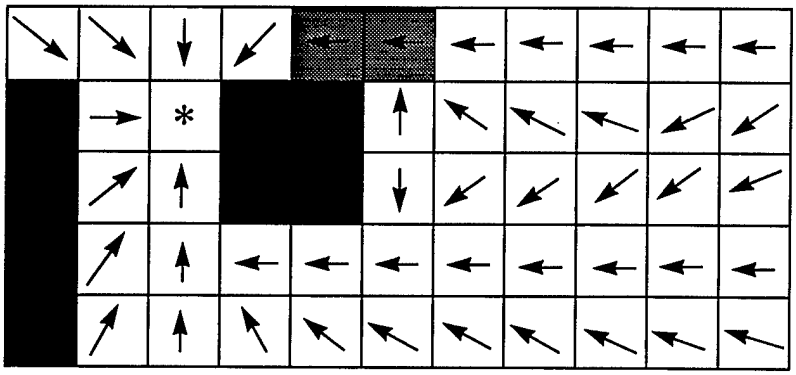


Figure 9: Example Trulla world map

4.0 Experiments and Results

This study was divided into two phases. The experiments in Phase I were designed to determine, for a changing environment, if the mean odometry error is comparable under both the learning and non-learning continuous localization techniques, and if the learning technique could provide accurate maps of the modified environment. In Phase II, continuous localization (with the learning behavior turned on) was combined with the Trulla path planner. The experiments for Phase II tested the performance of the composite robot system in various scenarios, and the results were compared to those of potential fields and Murphy's implementation of Trulla in similar situations.

4.1 Phase I

The first experiment established a mean odometry error for both the learning and non-learning localization techniques when the long-term map represented the true room configuration, i.e. there were no errors in the *a priori* long-term map. The experiment was composed of 8 runs for each technique, all with the same room configuration. Each run consisted of Coyote beginning at a randomly determined pose and then wandering the room randomly for fifty minutes, avoiding obstacles while continuous localization corrected its odometry. The randomness reduced the impact of the room configuration on the robot's ability to localize. The random wandering scheme was developed to maximize Coyote's coverage of the room. At one minute intervals all of Coyote's motors stopped to allow the robot to record its internal odometry and a human to physically measure the robot's true location. These paired pose readings allowed the error in the robot's odometry to be computed. The data obtained from this experiment was used as a control for the next experiment.

The second experiment tested each technique's ability to provide accurate localization when the *a priori* map significantly differed from the robot's true environment. Before each run, eight objects (such as chairs, desks, etc.) were moved in the real world, though their positions in the *a priori* map did not change. Each object was displaced thirty inches in a random direction from its original, true position and then rotated a random amount between -30 and +30 degrees from the original orientation. During each run the objects remained static. For each of eight distinct room configurations, one learning and one non-learning run were conducted. The learning rate was set to 10% (a weight of 0.1), and the random wandering scheme from the first experiment was used. Again, the robot was stopped each minute to record its internal odometry and true location.

The results of Phase I are summarized in Figure 10. Each data point in the graph represents the average of the eight runs for each of learning (dotted line) and non-learning (solid line) trials. The data points on the left show the average translational error when the true room did not deviate from the *a priori* map, and the data points on the right show the experiments where the room differed significantly from the *a priori* map.

As expected, in cases where the room had no changes, continuous localization with learning performed no better than the non-learning version (4.91 inches of translational error compared to 4.54 inches of error). The learning behavior assumed that the robot's pose was accurate, and it consequently altered the long-term map with inaccurate short-term perceptions.

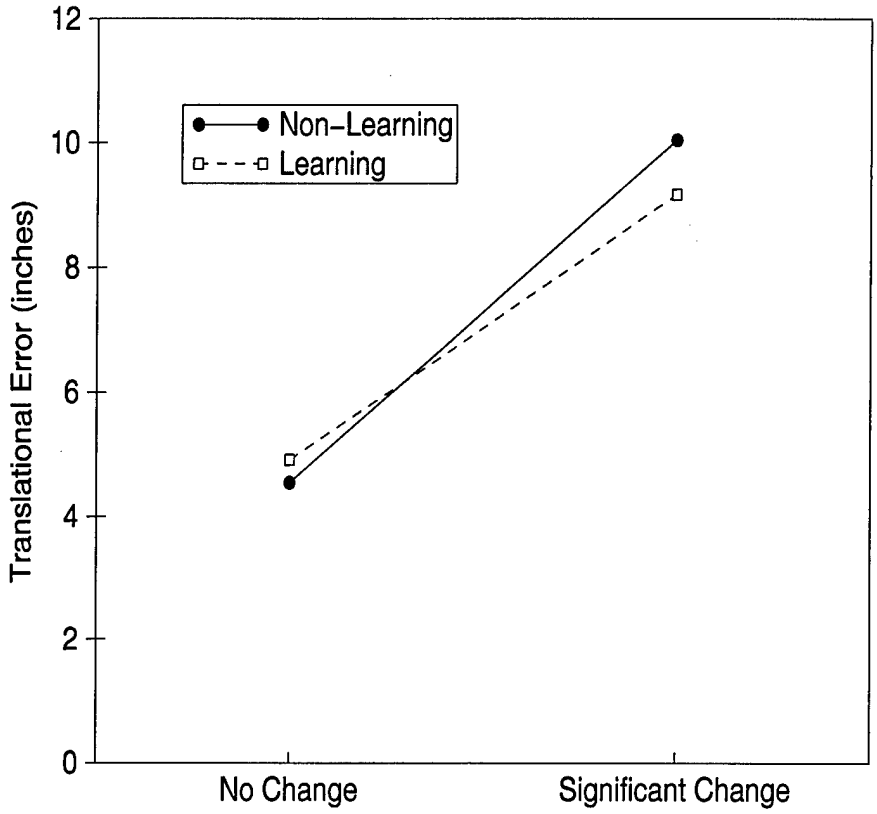


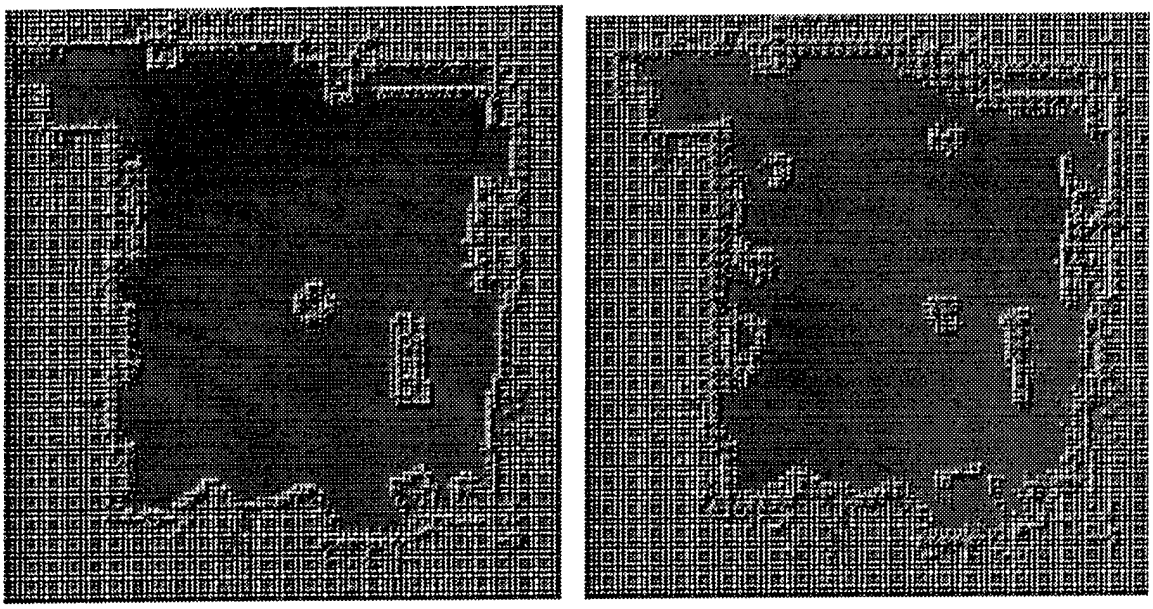
Figure 10: Amount of Change to Room

In the second experiment, where the room differed significantly from the map, the learning technique's mean odometry error of 9.29 inches performed marginally better than the non-learning result of 10.02 inches of error. Statistical analysis demonstrated that these two values correlated to the same number.

In addition to the odometry data collected, the long-term evidence grids used during the learning experiments were recorded at each update. These were used to produce an animation of the state of the long-term map while it was adapting to the environment. [1]

1. This animation can be seen at http://www.aic.nrl.navy.mil/~schultz/research/cont-local/animation.mov .

In Figure 11a, the initial frame of this animation is shown. This is accurate for a room with no difference from the original map, but is incorrect for the current, altered room. In Figure 11b, a frame from several minutes into the run shows how the map quickly converged to the true room configuration.



(a)
a priori map

(b)
Global map after 9 minutes

Figure 11: Continuous Localization (with the learning behavior)

4.2 Phase II

The second portion of this study demonstrated the parallel path planning and navigation structure implemented for Coyote (Figure 12). Continuous localization with the learning behavior ran beneath a deliberative/reactive architecture composed of Murphy's Trulla algorithm and the author's collision avoidance scheme. Trulla has been modified to use the maps generated by continuous localization's learning technique. This was achieved by converting the probability of occupancy of each evidence grid cell into a Trulla weight of traversibility.

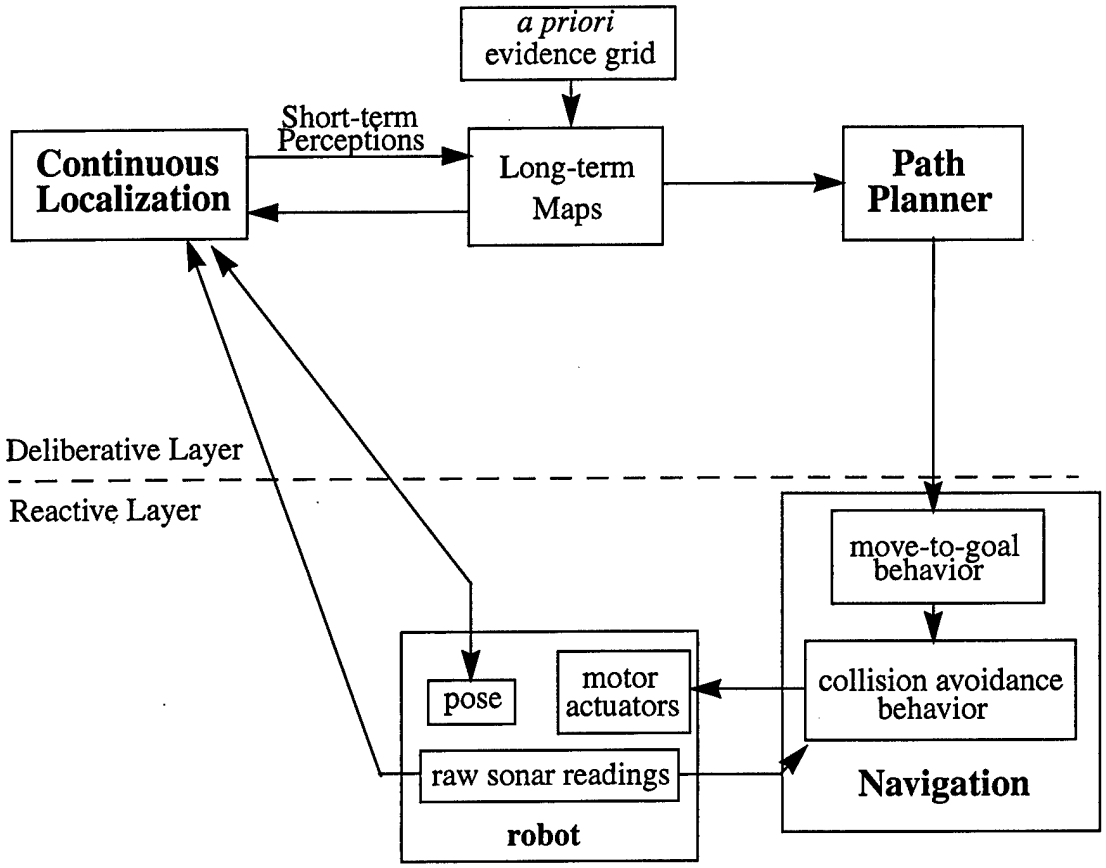


Figure 12: The Combined Robot System

Four significant differences exist between Murphy's and the author's system.

1) Grid Size. The author's system uses maps with a grid size of approximately five inches. The width of each square in Murphy's representation equals the diameter of the robot. (The diameter of Coyote is roughly two feet.) Thus the maps used by the author's system provide more detail.

Using a grid size equal to the robot's diameter provides the advantage that a path cannot be planned through a corridor too small for the robot to fit through. With grid cells only five inches wide, Trulla might plan the most optimal path through a space only five inches wide. Coyote would never get to the goal.

The author overcame this by artificially growing obstacles on the long-term map. The dimensions of each untraversible cell (i.e. walls and objects in the middle of the room)

were expanded by roughly the radius of Coyote and assigned a weight of 9. This reduced the probability of Trulla planning a path that would cause Coyote to collide with an obstacle.

Figure 13 shows a sample graphics window of the author's Trulla implementation. A green arrow is drawn in every empty cell. Although the arrow heads were removed to make the small line segments easier to distinguish, it is evident that they all point toward the goal (the red square). Yellow-green lines correspond to a weight of 4 (a mid-range probability on the evidence grid). The dark green lines, which are barely visible, are found adjacent to obstacles and the room's perimeter. They correspond to the area grown around untraversible space (a Trulla weight of 9). The yellow square labels the robot's current position, and the black trail behind the yellow square marks the path the robot has already traversed.

2) Continuous localization. The addition of continuous localization provided Coyote with an accurate update of its pose. Although both systems require an initial pose estimate, continuous localization allows the robot to run indefinitely without manually correcting the pose.

3) Update rate. Murphy's implementation replans when a complex set of criteria are met that indicate the robot has significantly deviated from the optimal path. The implementation presented in this study replanned when a new long-term map was created by continuous localization. Continuous replanning would not have provided an advantage since new evidence was not available until a new long-term map was generated by the learning behavior.

4) Robot speed. Murphy's robot moved at only two inches per second, an extremely slow speed for mobile robots. If an unmodeled obstacle existed in the robot's path 20 inches away, her robot would have 10 seconds to acquire data and replan a path around the object. Coyote, which moved at 10 inches per second, would only have 2 seconds to avoid an object at the same distance. With the increased reaction time, Murphy's robot system could plan a more-optimal path in the presence of unmodeled obstacles. However, the system this study presents would travel to the goal in less overall time.

During the experiments for Phase II, continuous localization's learning rate was set at .5. Continuous localization generated an updated map after every 12 inches of robot translation.

Trial 1 tested the robot system's ability to explicitly plan a path around an unmodeled box canyon. The *a priori* map (Figure 15a) did not model the crates that blocked Coyote's direct route to the goal (Figure 15b). For each of ten runs, Coyote began from an identical position with a random orientation. The initial position bisected the obstacle. When Coyote's collision avoidance stopped the robot from hitting the wall blocking its path, equal probability existed to navigate left or right around the obstacle.

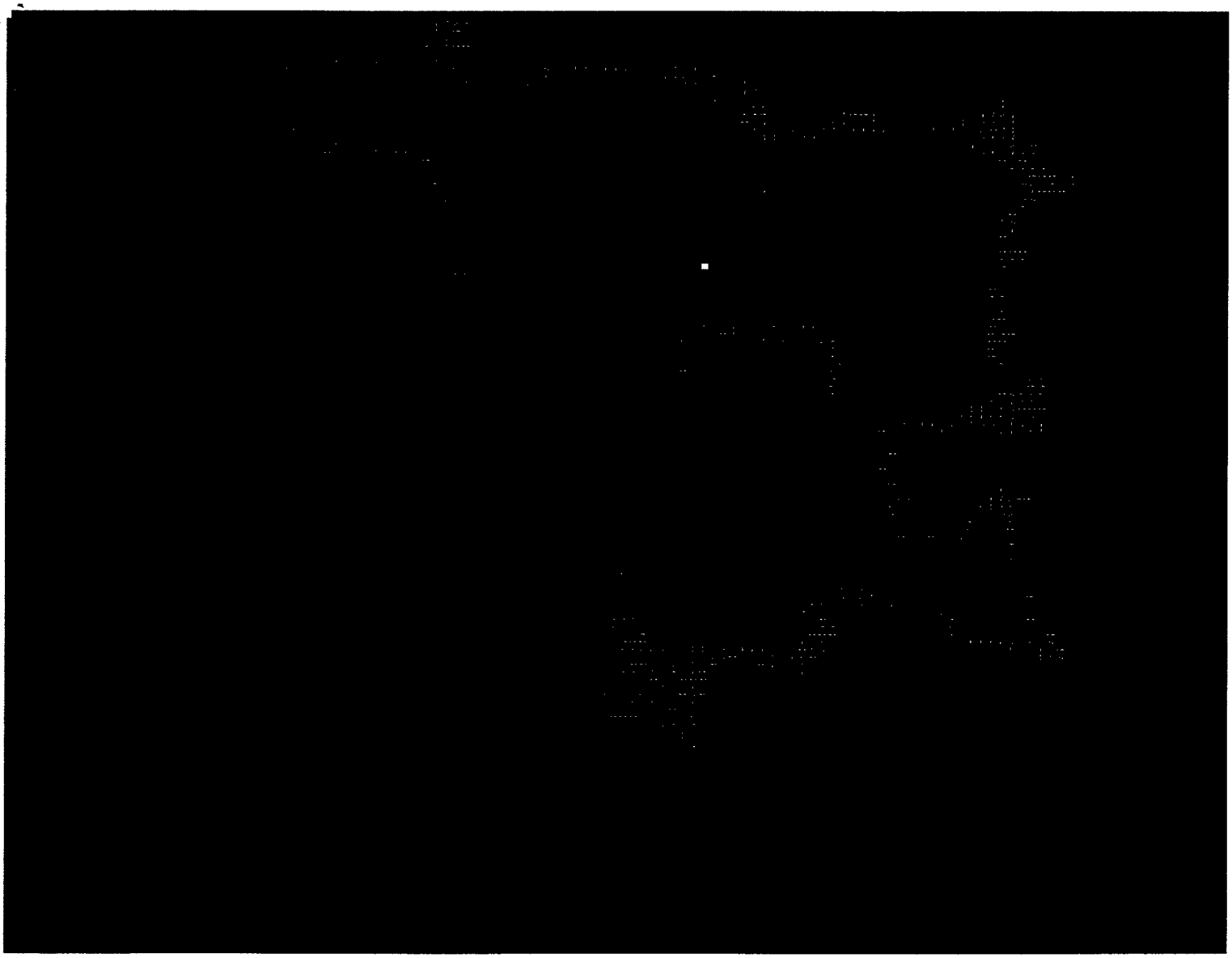


Figure 13: An Example Path Planner Graphics Window

The results from the first trial are displayed in Figure 14. Coyote navigated to the goal at an average of 104.5 seconds. During each run, the path planner guided Coyote into the box canyon. Once it reached the perpendicular wall, Coyote oscillated until its sensors detected the walls and a new long-term map was generated. The long-term evidence map only learned the perimeter of obstacles directly in Coyote's path (Figure 15c).

Coyote established an optimal time of 52.1 seconds when it traveled from the goal to the initial position with an *a priori* map that exactly modeled the room configuration.

Run	Time (sec.)	Number of Updates
1	103.0	16
2	95.9	16
3	95.0	15
4	117.0	20
5	114.3	20
6	128.5	22
7	93.8	15
8	101.7	16
9	97.5	15
10	98.0	15
Average	104.5	17

Figure 14: Time to Travel out of Box Canyon (Phase II, Trial 1)

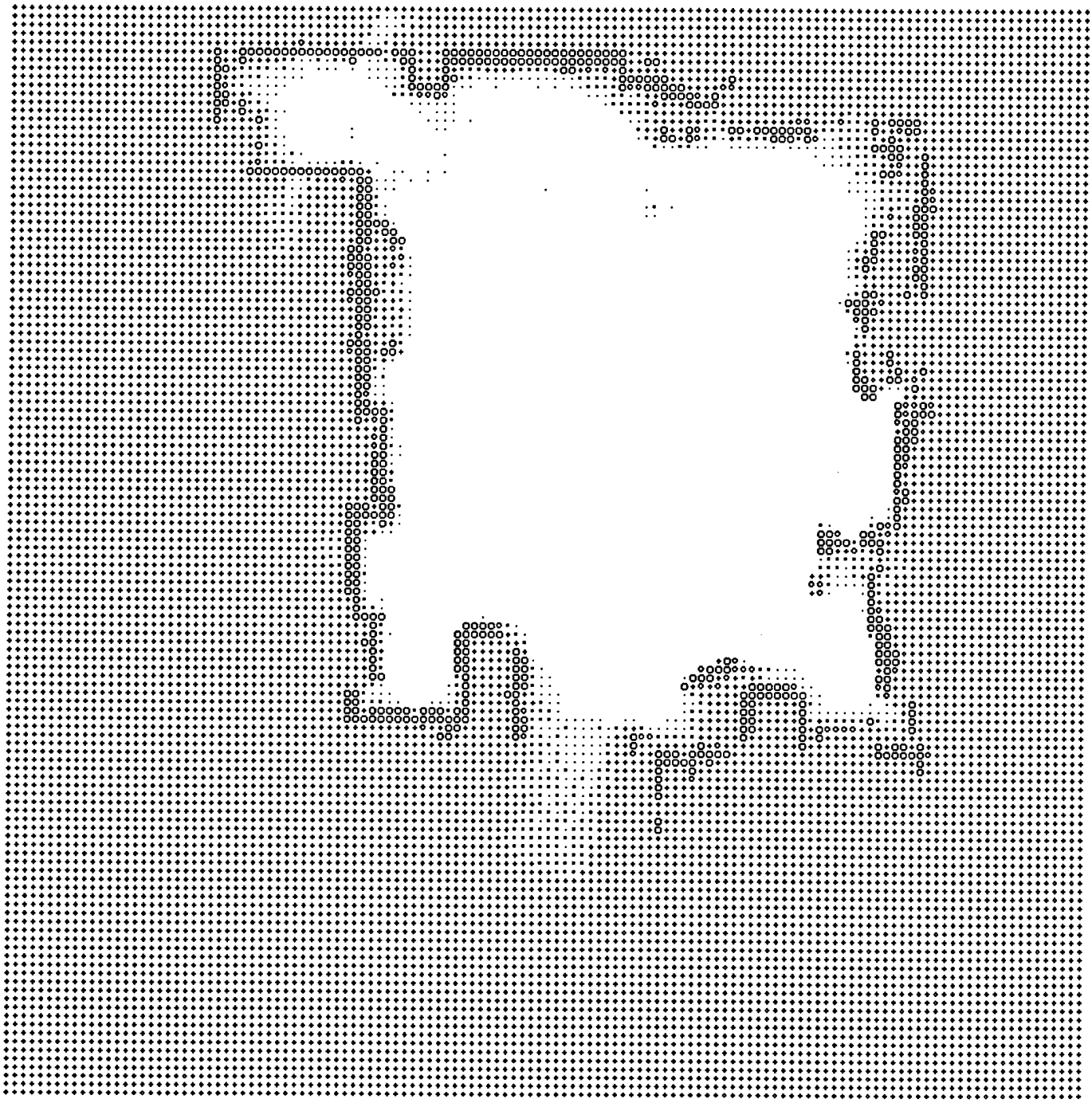


Figure 15a: *A Priori* Map (Phase II, Trial 1)

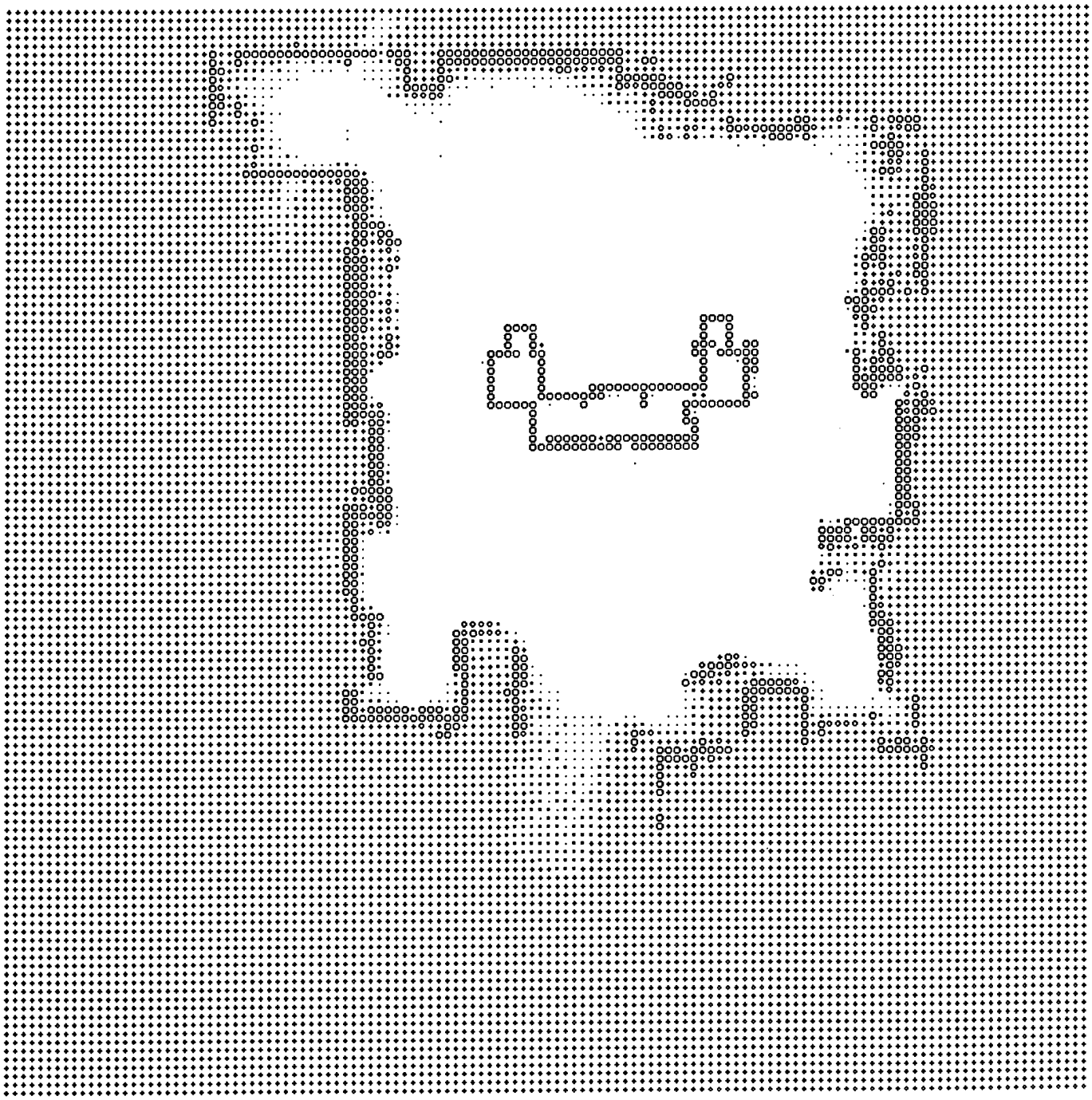


Figure 15b: Actual Room Configuration with Box Canyon (Phase II, Trial 1)

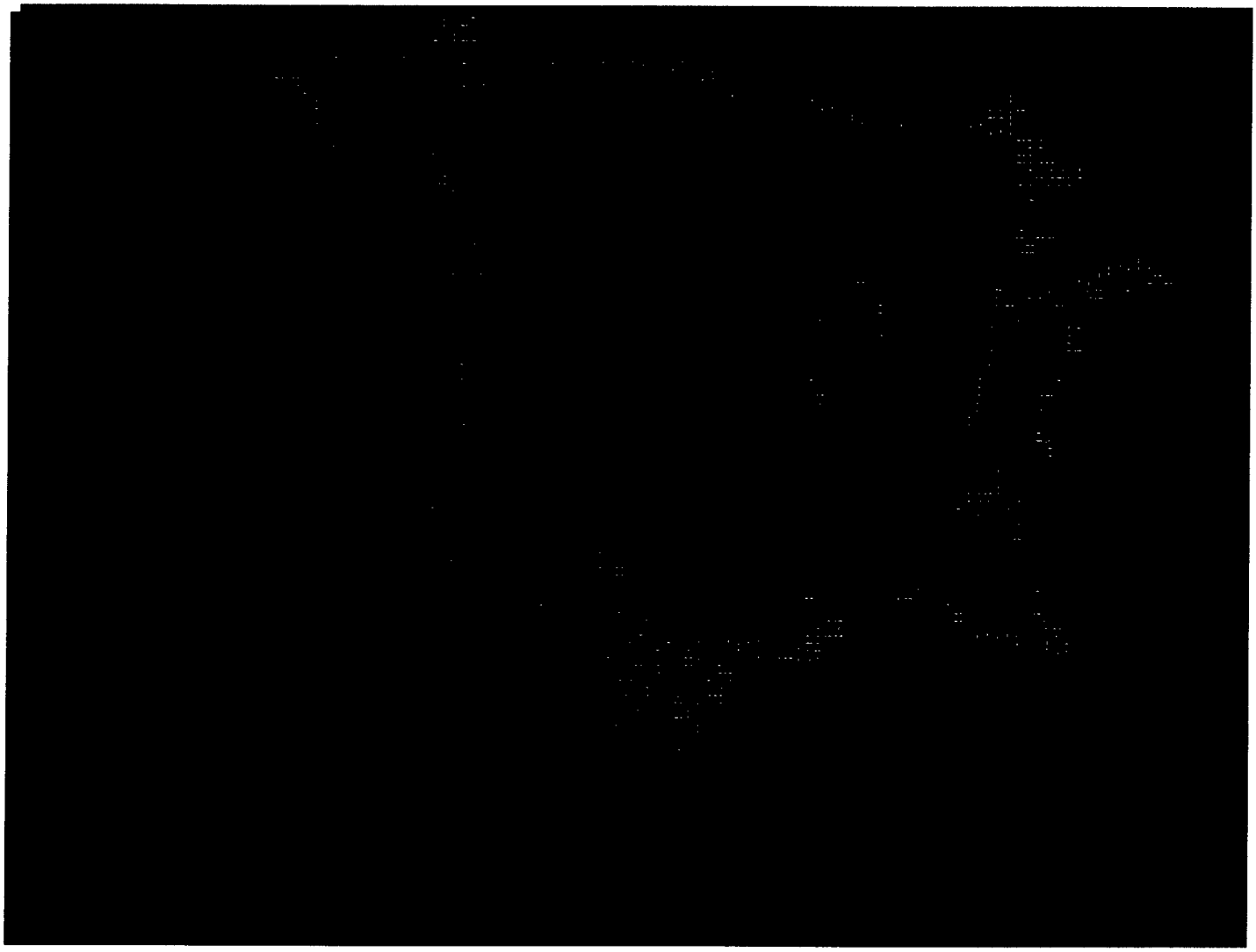


Figure 15c: Continuous Localization Learned the Box Canyon (Phase II, Trial 1)

Trial 2 tested the robot system's ability to navigate through a passageway slightly wider than Coyote's diameter (Figure 16). The collision avoidance was modified to ignore sonar readings (which would not have allowed the robot to within 15 inches of an object). The sonars were still used for continuous localization. The corridor measured 13 inches wider than Coyote's diameter. Coyote's speed was set at 10 inches per second.

The repulsive forces from both walls simultaneously acted upon the robot. By avoiding the right wall, Coyote steered left and arrived in a cell that directed it back to the right, and vice versa. This oscillating motion severely slowed Coyote's progress. The robot navigated the twenty foot corridor in 196.6 seconds (an average of 1.2 inches per second).

After determining that Coyote could navigate a narrow corridor, albeit slowly, the third trial tested the robot system's ability to detect an unmodeled, more efficient route to the goal. An object was modeled *a priori* (Figure 17a). However, a portion of the object was removed before runtime to allow a short cut to the goal (Figure 17b). Twenty runs were conducted. Coyote began and ended each run bisecting the gap at a distance of four feet from the crates. The goal location of the previous run became the initial position of the following run, and continuous localization updated maps throughout the entire trial.

The robot system did not navigate Coyote through the gap. A few of the long-term maps showed that a gap existed between the two crates. However, before the robot system replanned using one of these maps, more recent sonar readings caused the learning scheme to think that the corridor had been sealed off. This occurred because Coyote moved away from the gap too quickly. When the robot was directly between the two crates, a sonar beam travelled the entire length of the gap and the system learned the hole. However, when a sonar entered the gap at an angle, it would strike one of the crates. When the system updated every cell within the volumetric cone, enough occupied evidence existed in the actually empty cells that the robot system interpreted the gap to be closed.

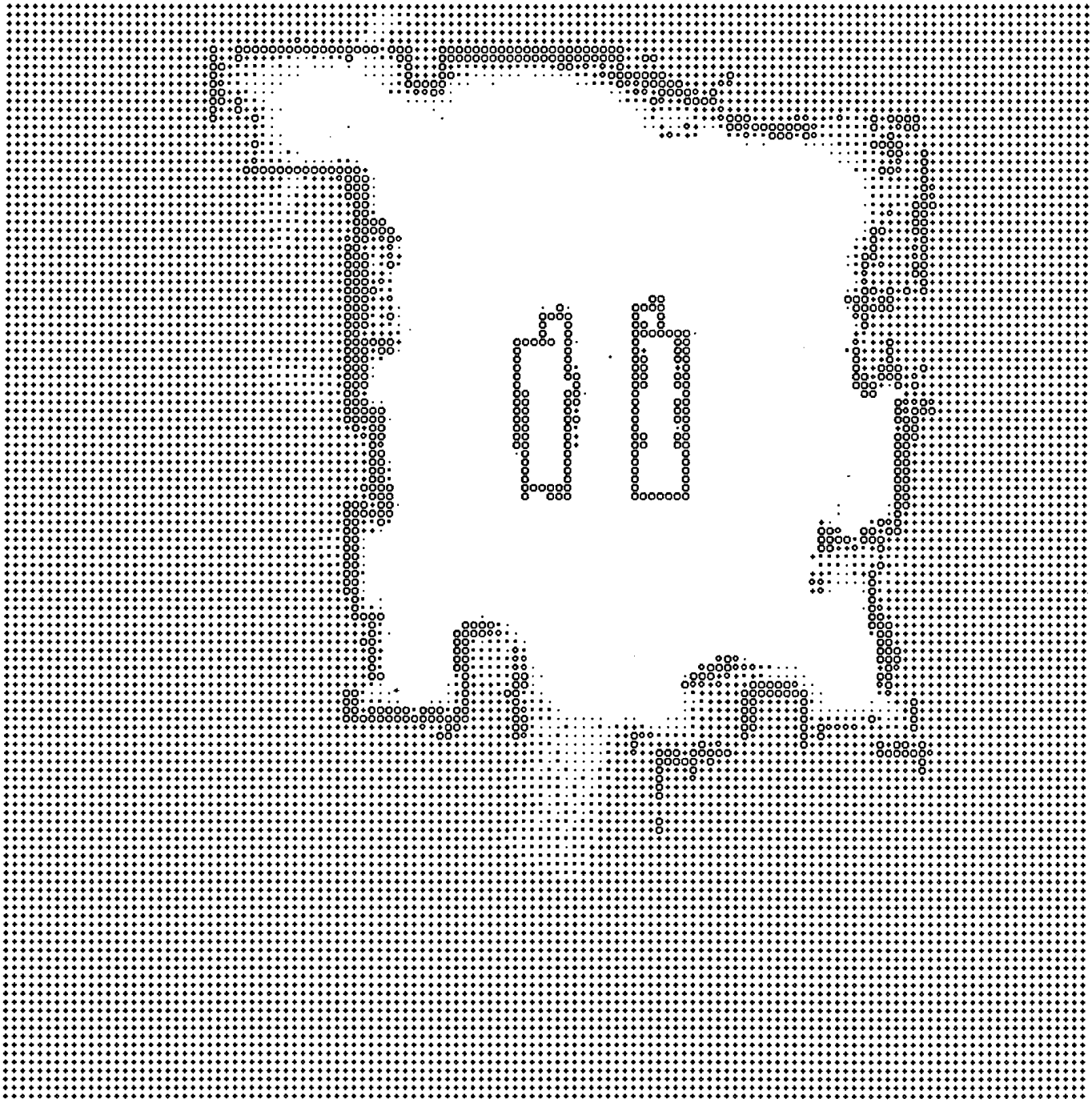


Figure 16: Narrow Corridor (Phase II, Trial 2)

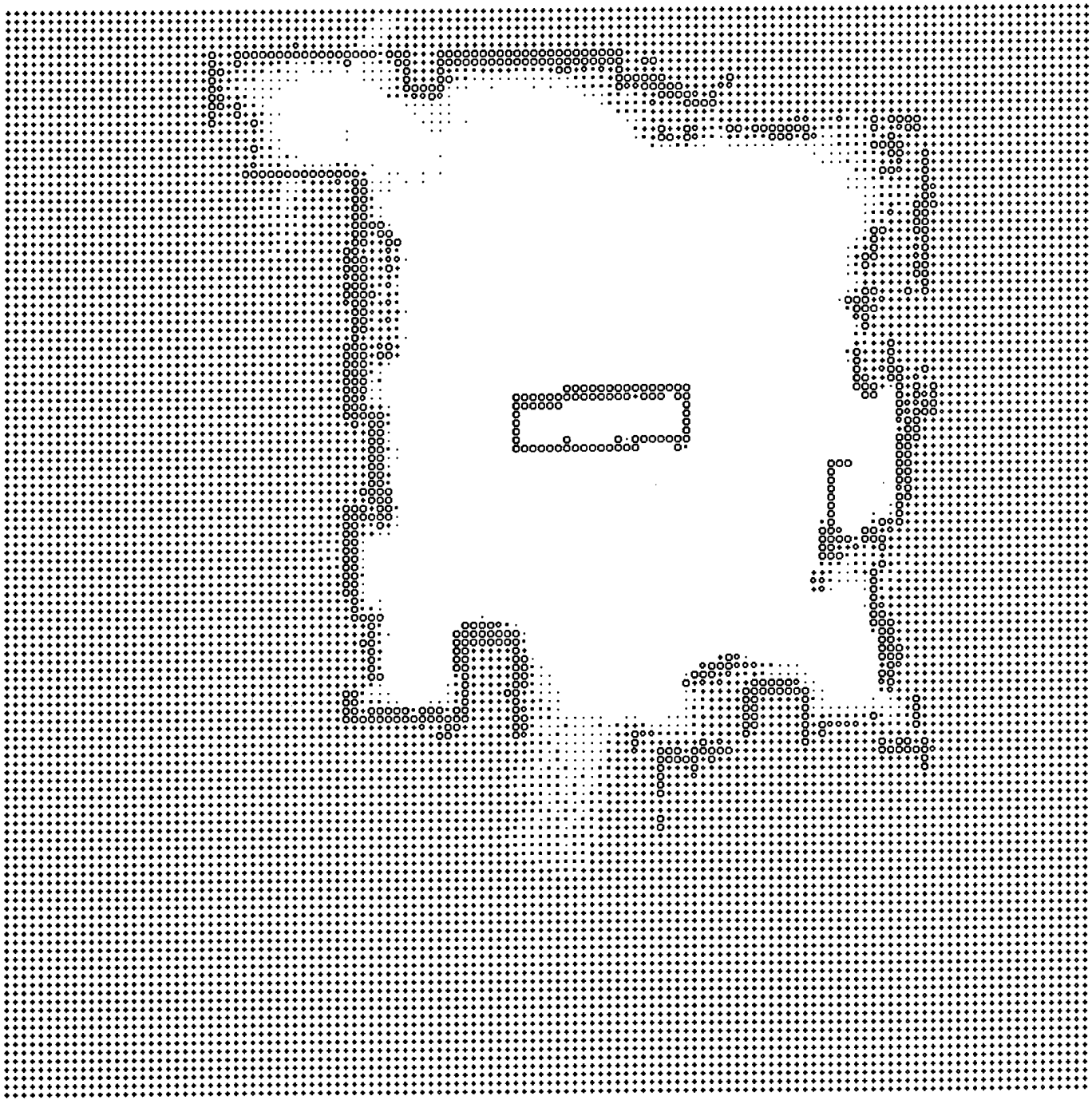


Figure 17a: *A Priori* Map did not Model the Gap (Phase II, Trial 3)

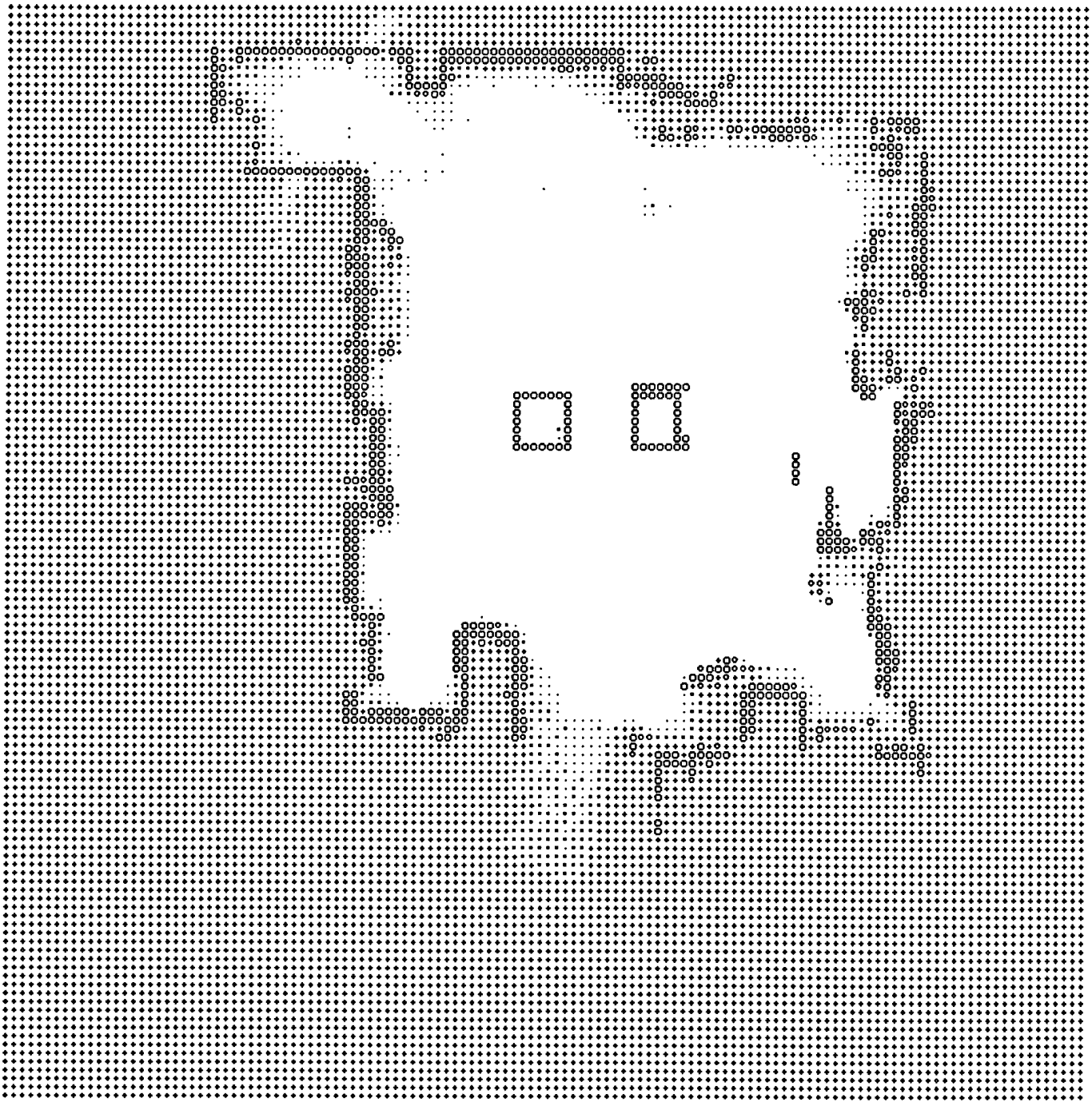


Figure 17b: Actual Room Configuration with Gap (Phase II, Trial 3)

5.0 Discussion

Three observations can be made regarding the results obtained from Phase I. First, the original, unmodified continuous localization technique performed much better than anticipated when the true room deviated from the *a priori* map used for localization. This robustness seems to be due to the way registration was performed, which tended to ignore differences between the maps and concentrated on similar regions. Note that in these experiments, furniture was moved, but not all of the furniture nor the walls of the room. If the walls of the room were moved, a much larger corresponding error in the non-learning localization would be expected.

A second observation is that with learning, the map quickly adapts to the changes in the environment and learns the correct room layout. In the animations, it is possible to see rapid changes in the evidence grid as the map adapts due to the new sensor readings.

The third observation, noticed from the animations, is that after a long period of updating the odometry, some noise would start accumulating in the map. This effect tended to be noticed only after long periods of updating, such as 30 to 60 minutes. This noise is produced as a side effect of the method used to update the long-term map from the short-term map, and because the maps still have a constant error of up to 5 inches. However, this does not affect continuous localization's ability to remain accurately localized, as seen in these results.

The second phase confirmed that a solution of the robot problem necessitates all four low-level autonomous mobile robot abilities to be truly robust. However, a simple collision avoidance scheme proved ample for environments without narrow, navigable passageways.

Trial 1 demonstrated that this study's robot system achieved its objectives. The evidence grids generated by continuous localization were used by the path planner to create a vector field that provided the robot with an optimal direction to the goal from any position on the map. The learning component of continuous localization updated the *a priori* map by fusing local sensor data with the long-term map. The robot system replanned optimal routes on the fly while the robot moved at 10 inches per second. The replanning time varied between .5 and 1.5 seconds. This compares favorably with Murphy's results. The evidence grids created by continuous localization contain 16,384 cells. Murphy's grid of the same area would hold merely 712 cells.

After replanning, Coyote navigated to the goal while maintaining accurate pose. Minor errors have little impact on the navigation of a short distance. Odometry error will not accumulate rapidly. But in long-distance travel, small changes to the *a priori* map that remain uncorrected will cause the robot to become lost. The experiments in Phase I determined that continuous localization will maintain an average error of five inches indefinitely, even when the *a priori* map differs significantly differed from the actual surroundings. In a previous experiment with non-learning continuous localization, Coyotes began a run 50 inches away from the initial position it was told it was at, and recovered to within five inches of error.

Occasionally, instead of reaching the goal, Coyote entered an orbit around it. The path planner was not at fault. Rather, the problem occurred when the motor commands given to Coyote did not rotate the robot quickly enough. Before Coyote could face the goal, it had moved to an adjacent cell. At such a close proximity to the goal, the angles to the goal from adjacent cells differ immensely. When stuck in this situation, the robot wanders in circles indefinitely because it never aims toward the goal.

Variable velocity commands would alleviate this problem. Increasing the rotational velocity and decreasing the translational velocity as the robot approaches the goal would cause Coyote to follow a direct path to the goal. This compares to an automobile driver easing off of the brake pedal as the car comes to a stop.

The second and third trials attested to the poor performance of a purely-reactive collision avoidance scheme in a situation that demanded fine-tuned navigation. In Trial 2, Coyote oscillated along a winding path instead of traveling directly between the walls (as the path planner suggested). In Trial 3, when Coyote's sensors detected an open corridor, the robot system failed to lead Coyote along this short cut.

NCARAI is developing a collision avoidance scheme which would capitalize on the situations of the second and third trials (Yamauchi et al., 1997a). It divides open space into corridors wide enough for the robot and then chooses the corridor that provides an optimal path to the goal.

6.0 Conclusions

The first phase of this study compared a learning and a non-learning technique for continuous localization. Both continuous localization techniques maintained a constant mean translational error over time. Although the performance of the two techniques was essentially identical, the learning technique offers a tremendous advantage over its non-learning counterpart: a map that accurately represents the surrounding region.

An open question derived from this study is, if learning is left operational over even longer periods of time, will the noise in the learned map render it unusable for localization? NCARAI is investigating several ways to address this. First, it is possible to look at the actual updates made over time to the encoders, and learn a model of the error. In the case of systematic error, this model can be used to compensate for the residual error that would result from direct application of the correction. Removing this residual error before updating the map should allow the map to be updated for many hours with no appreciable accumulation of error in the map.

Second, results suggest that updated maps from continuous localization's learning not be saved as the true map of the room. Current research at NCARAI combines a map-building exploration strategy with the continuous localization (Yamauchi, 1997). When the robot enters an unexplored region, it builds a new map, and then stores that map for that room. When the robot enters that room again, the stored map is used, along with the adaptive continuous localization technique. The updated map is used while in the room for navigation and reasoning about the spatial characteristics of the room, but when the robot exits the room and reenters, the original map is used, without the updates. This should work well because the continuous localization with learning quickly learns new features of the room.

The results from Phase II demonstrated that the Trulla algorithm, when combined with continuous localization and a collision avoidance scheme, provided a competent deliberative/reactive navigation system. However, the system performed poorly in narrow passageways due to its simple collision avoidance.

NCARAI currently uses this study's robot system. A future addition will include the creation of a separate collision avoidance scheme that resides on Coyote and works in conjunction with any motion-based process. The author augmented current work at NCARAI by developing a method for mapping a room that does not require giving the robot an initial pose (Yamauchi et al., 1997b). After the robot performs this mapping task, the resulting map will become the *a priori* map for continuous localization and the path planner.

Coyote's inability to learn unmodeled empty space raises an interesting question. Should a mobile robot system learn empty and occupied space at different rates?

Coyote's structured light range finder quickly identifies unmodeled occupied space. Its probability profile equations attribute a great deal of certainty to a reading. (No readings are available when the laser plane strikes outside of the camera's view.) How-

ever, transient objects, such as a person briefly walking past a robot, should not be recorded on the long-term map. Perhaps unmodeled occupied space should be learned slowly.

Unmodeled empty space represents an object removed since the creation of the *a priori* map, such as furniture. If this object is unlikely to reappear, unmodeled empty space should be assimilated into the long-term map quickly.

Current robot systems share few abilities with science fiction robots. Mobile robots soon will accomplish high-level tasks without human oversight. This study contributed toward a universal solution to the robot problem by incorporating a learning continuous localization behavior and a robust path planner.

The work in this study has produced two papers for the author (Graves et al., 1997; Yamauchi et al., 1997b). The latter paper has been accepted for presentation at the Fourteenth National Conference on Artificial Intelligence (AIII-97). The former paper was accepted to the IEEE International Symposium on Computational Intelligence in Robotics and Automation. The author will present the paper in Monterey, California, on July 10, 1997.

7.0 Acknowledgments

My sincere thanks are extended to everyone who helped with this Trident research project. Dr. Patrick Harrison, Deputy Chairman of the U.S. Naval Academy Computer Science Department, provided firm guidance throughout the duration of the project. The expertise of Dr. John Grefenstette, Alan Schultz, and William Adams -- the members of the robotics lab at the Navy Center for Applied Research in Artificial Intelligence located at the Naval Research Laboratory -- was indispensable. The author also appreciates Associate Professor Wayne Ehler, U.S. Naval Academy Mathematics Department, for his patience in explaining applied statistics. All of your contributions are very much appreciated.

8.0 References

Arkin, R., (1989). "Motor Schema-Based Mobile Robot Navigation," *International Journal of Robotics*, Vol. 8, No. 4, MIT Press: Cambridge, MA, August 1989, pp. 92-122.

Chung, H., Choi, Y. and Lee, J., (1992). "Path Planning for a Mobile Robot with Grid-Type World Model," *Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent* Robots and Systems, Raleigh, NC, July 7-10, 1992, pp. 439-444.

Elfes, A., (1987). "Sonar-Based Real-World Mapping and Navigation," *IEEE Journal of Robotics and Automa*tion RA-3, 1987, pp. 249-265.

Elfes, A., (1992). "Dynamic Control of Robot Perception Using Multi-Property Inference Grids," *Proceedings of the IEEE International Conference on Robotics and Automation*, Nice, France, May 1992, pp. 2561-2567.

Graves, K., Schultz, A., and Adams, W., (1997). "Continuous Localization in Changing Environments," to appear in *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, Monterey, CA, July 10-11, 1997.

Horn, J. and Schmidt, G., (1995). "Continuous Localization of a Mobile Robot Based on 3D-laser-range Predicted Sensor Images, and Dead-Reckoning," *Robotics and Autonomous Systems*, Elsevier Science, 1995, pp. 99-118.

Hughes, K. and Murphy, R., (1992a). "Ultrasonic Robot Localization using Dempster-Schafer Theory," *SPIE Stochastic Methods in Signal Processing, Image Processing, and Computer Vision*, invited session on Applications for Vision and Robotics, San Diego, CA, July 19-24, 1992.

Hughes, K., Tokuta, A., Ranganathan, R., (1992b). "Trulla: an Algorithm for Path Planning Among Weighted Regions by Localized Propagations," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Raleigh, NC, July 7-10, 1992.

Koren, Y. and Borenstein, J., (1991). "Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation," *Proceedings of the IEEE International Conference on Robotics and Automation, Sacramento*, California, April 9-11, 1991, pp. 1398-1404.

Meystel, A., (1991). *Autonomous Mobile Robots: Vehicles with Cognitive Control.* Teaneck, N.J.: World Scientific Publishing, 1991.

Moravec, H., (1981). *Robot Rover: Visual Navigation.* UMI Research Press: Ann Arbor, MI, 1981.

Moravec, H. and Elfes, A., (1985). "High Resolution Maps from Wide Angle Sonar," The Robotics Institute, Carnegie-Mellon University, 1985.

Moravec, H., (1988). "Sensor Fusion in Certainty Grids for Mobile Robots," *AI Magazine*, Vol. 9, No. 2., Summer 1988, pp. 61-74.

Moravec, H. and Cho, D.W., (1989). "A Bayesian Method for Certainty Grids," The Robotics Institute, Carnegie-Mellon University, 1989.

Moravec, H. and Blackwell, M., (1992). "Learning Sensor Models for Evidence Grids," In *Robotics Institute Research Review*, Pittsburgh, PA, 1992.

Murphy, R., Hughes, K., Marzilli, A., and Noll, E., (1997a). "Integrating Explicit Path Planning with Reactive Control of Mobile Robots," internal paper, 1997.

Murphy, R., Marzilli, A., Hughes, K., (1997b). "When to Explicitly Replan Paths for Mobile Robots," submitted to *International Conference on Robotics and Automation*, Albuquerque, NM, April 20-25, 1997.

Nilsson, N. (ed.), (1984). *Shakey the Robot.* SRI International. Technical Note 323, April 1984.

Schultz, A. C., Adams, W., and Grefenstette, J. J., (1996). "Continuous Localization Using Evidence Grids," NCARAI Report AIC-96-007, Naval Research laboratory, Washington, D.C., 1996.

Thrun, S., (1996). "A Bayesian Approach to Landmark Discovery and Active Perception in Mobile Robot Navigation," *Proceedings of the IEEE International Conference on Robotics and Automation*, New York, NY: IEEE, 1996, pp. 1401-1406.

Yamauchi, B. and Langley, P., (1997a). "Place Recognition in Dynamic Environments," *Journal of Robotic Systems*, Special Issue of Mobile Robots, Vol. 14, No. 2, February 1997, pp. 107-120.

Yamauchi, B., Schultz, A., Adams, W., Graves, K.,Grefenstette, J., and Perzanowski, D., (1997b). "ARIEL: Autonomous Robot for Integrated Exploration and Localization," to appear in *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, 1997.

Appendix A

Probability Profiles of a Volumetric Sonar Cone

The following equations were devised by Moravec and Elfes to model the behavior of ultrasonic waves.

"Consider a position P = (x,y,z) belonging to the volume swept by the sonar beam. Let:

R be the range measurement returned by the sonar sensor,

ε be the mean sonar deviation error,

ω be the beam aperture,

$S = (x_s, y_s, z_s)$ be the position of the sonar sensor,

δ be the distance from P to S,

θ be the angle between the main axis of the beam and SP." (Moravec, 1984)

Moravec's Probability Profiles graphically demonstrate the two regions of the sonar beam (Figure A-1). The empty region represents the points inside the beam $(\delta < R - \varepsilon \text{ and } \theta \leq \omega/2)$ that are probably empty. The somewhere occupied region correlates to the points encircling the front of the cone $(\delta \in [R - (\varepsilon, R + \varepsilon)] \text{ and } \theta \leq \omega/2)$ that account for the inaccuracy of the range. Both probabilities (p_E and p_O) are functions of distance and direction.

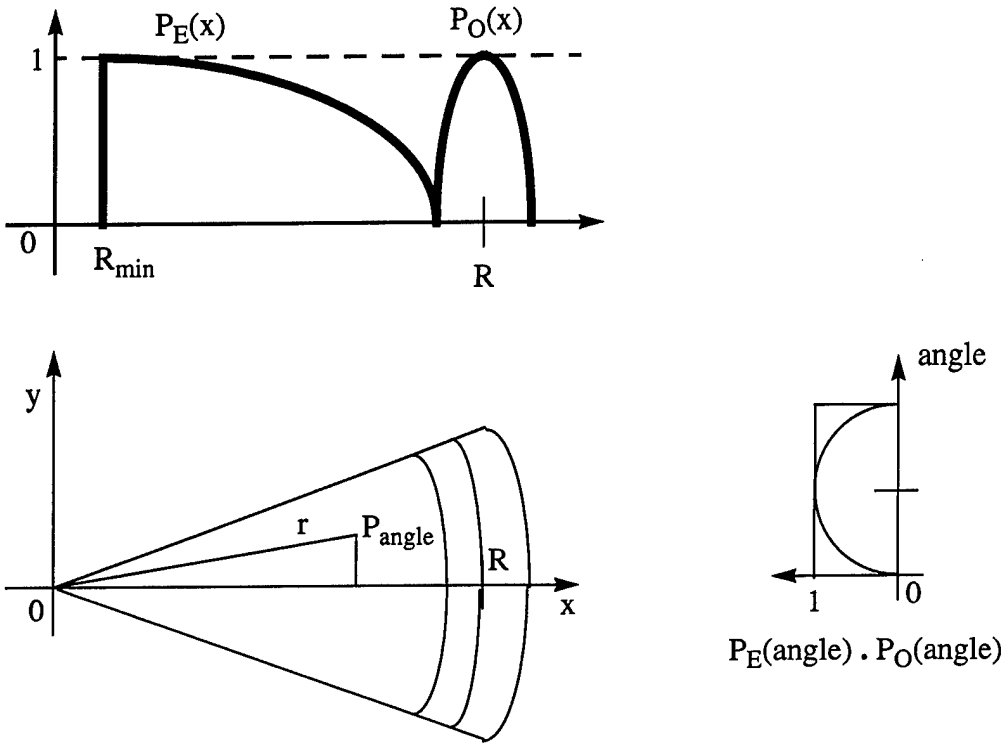


Figure A-1: Probability Profiles (Moravec, 1984)

The functions were then applied to the preprocessed data. The empty probability distribution function is:

$$p_E(x,y,z) = E_r(\delta) \cdot E_a(\theta)$$

where: $E_r(\delta) = 1 - ((\delta - R_{min})/(R - \varepsilon - R_{min})^2)$ for $\delta \in [Rmin, R - \varepsilon]$

$E_r(\delta) = 0$ otherwise

and: $E_a(\theta) = 1 - (2\theta/\omega)^2$ for $\theta \in [-\omega/2, \omega/2]$

The occupied probability density function is denoted by:

$$p_O(x,y,z) = O_r(\theta) \cdot O_a(\theta)$$

where: $O_r(\theta) = 1 - (\delta - R)/\varepsilon)^2$ for $\delta \in [R - \varepsilon, R + \varepsilon]$

$O_r(\theta) = 0$ otherwise

and: $O_a(\theta) = 1 - (2\theta/\omega)^2$ for $\theta \in [-\omega/2, \omega/2]$

The two probability distribution functions are not identical because they model two distinct regions. The empty area is assumed to be empty. However, an uncertainty is associated with the occupied area, especially around the edges of the readings.

Appendix B

Bayesian Statistics

The data from each sonar reading was entered into these functions, and then the results were normalized and combined using Bayes' theorem. A brief review of probability will aid in understanding Bayes' theorem. Let $p(A|B)$ represent the probability (estimated degree of certainty) of situation A given that information B is true, and $p(A)$ represent the likelihood of situation A given no information. More specifically, $p(o_{(x,y)}|M)$ relates the probability that cell (x,y) is occupied given the sonar data M, and, when speaking of only one particular cell, $p(o)$ relates the probability that that cell is occupied when no information is given. Also, $\bar{A}$ refers to the alternative of A. Then the following axioms hold true:

$p(A) \leq 0$ for all A A1

$p(A|A) = 1$ for all A A2

$p(A|B) + p(\bar{A}|B) = 1$ A3

$p(A \wedge B) = p(A|B)p(B)$ or $p(A|B) = \frac{p(A \wedge B)}{p(B)}$ A4

Moravec expressed Bayes' theorem as the following:

$$\frac{p(o|M)}{p(\bar{o}|M)} = \frac{p(M|o)}{p(M|\bar{o})} \times \frac{p(o)}{p(\bar{o})}$$ (Moravec, 1989)

Moravec chose this representation because the independent sources of information o and M are combined into a single estimate: $p(o|M)$. However, Moravec realized that trying to produce a map from new information M_1 and M_2 after each had been processed into a map *individually* (i.e. determine $p(o|M_1 \wedge M_2)$ given $p(o|M_1)$ and $p(o|M2)$), would be trying to determine $p(A \wedge B)$ given $p(A)$ and $p(B)$. There simply was not enough information. But by assuming the independence of A and B, Moravec could express the following:

$p(A \wedge B) = p(A)p(B)$ instead of using A4. This yielded the combining equation:

$$\frac{p(o|M_1 \wedge M_2)}{p(\bar{o}|M_1 \wedge M_2)} = \frac{p(o|M_1)}{p(\bar{o}|M_1)} \times \frac{p(o|M_2)}{p(\bar{o}|M_2)}$$

If a robot remains stationary then the sonars will return the same information for (at least) two consecutive readings, even though they were taken from the same pose and therefore do not add new information to the map. In mathematical terms, M_1 and M_2 are not entirely independent. They share common information M_c. Moravec's combining equation then became:

$$\frac{p(o|M_1 \wedge M_2)}{p(\bar{o}|M_1 \wedge M_2)} = \frac{p(o|M_1)}{p(\bar{o}|M_1)} \times \frac{p(o|M_2)}{p(\bar{o}|M_2)} / \frac{p(o|M_c)}{p(\bar{o}|M_c)}$$

Moravec used $p(o|M_1)$ to represent the evidence grid that was being updated and $p(o|M_2)$ as a reading (Moravec, 1989). Note that this method allows a robot to fuse data from all of its sensor inputs, not just sonar.

After Moravec's combining equation was applied to the sonar data, p_E and p_O were added to each cell on the map. Each time that the grid was updated the two resulting numbers in each cell were then compared against each other to yield a final value:

$$Map(X,Y) := \begin{cases} Occ(X,Y) & \text{if } Occ(X,Y) \geq Emp(X,Y) \\ -Emp(X,Y) & \text{if } Occ(X,Y) < Emp(X,Y) \end{cases}$$

Thus Max(X,Y) is a number on the interval [-1, 1]. This result is added to the value currently held in cell (X,Y) to yield the updated evidence of occupancy.